E.R.G. SYSTEMS, INC.

BRACKENWOOD PATH • HEAD OF THE HARBOR, NEW YORK 11780 • (516) 584-5540

JOHN L. REMO, PH.D.
PRESIDENT

Final Report

Full Surface Testing of Grazing Incidence Mirrors

September 12, 1992

John L. Remo Ph.D., Principal Investigator

Research is supported by SDIO/IST and managed by ONR
Contract # N00014-90-C-0246
Contract period: 9/12/90 - 9/12/92

TABLE OF CONTENTS

1.0	**INRODUCTION**	1
2.0	**PRINCIPLES AND ANALYSIS**	2
3.0	THE OPTICAL INTERFEROMETER SUBSYSTEM	3
3.1	System Operation	3
3.2	The Optical Prototype	6
4.0	**DIGITAL AND ELECTRONICS HARDWARE**	13
4.1	Control Electronics Hardware	15
4.1.1	Scanning Stage Control	15
4.1.2	Microstepper Stage Control	15
4.2	Data Acquisition Hardware	23
4.2.1	The CCD Camera	23
4.2.2	The Frame Grabber/Buffer Board	24
4.2.3	Output Devices	24
4.3	Data Processing Hardware	26
5.0	**SOFTWARE**	27
5.1	Software Overview	27
5.2	User Manual	28
5.3	Software Package Source Code	30
6.0	**TESTING AND EXPERIMENTAL RESULTS**	72
6.1	The Test Mirror	72
6.2	Subaperture Interferograms	72
6.3	Subaperture Testing: Detailed Operation	73
6.4	Subaperture Testing: Final Results	73

FULL-SURFACE INTERFEROMETRIC SCANNER FOR THE MEASUREMENT OF GRAZING INCIDENCE MIRRORS

1.0 INTRODUCTION

The objective of the Phase-II research was to design, construct and test a prototype interferometer system (FSIS) for the purpose of testing cylindrical-like mirrors such as those used in grazing incidence X-ray imaging applications.

Existing instruments are based on a shearing interferometry, dual laser beam scanning approach, measuring the slope or curvature functions along a **single long trace** in the long direction of the cylindrical mirror. While these instruments are characterized by a high measuring accuracy, they are very slow and environmentally sensitive. An even slower and more complex two-dimensional scan must be implemented if the full surface measurement is required.

Our approach is also based on interferometric slope measurement in the direction of the long side of the mirror, using a grating shearing interferometer head. A full-width beam covering an extended subaperture of the mirror is scanned in discrete steps in the direction of the long side of the mirror, so that a single scan allows coverage of the whole mirror surface. By least square fitting the series of subaperture measurements, the system yields parallel measurement of a **multiplicity of long traces** in the direction of scan.

In the following final report we describe the work on the prototype FSIS system including a description of

- The optical interferometer subsystem
- The digital and electronics hardware
- The data reduction software.

We then describe the testing program and present experimental results for a cylindrical test surface.

2.0 **PRINCIPLES AND ANALYSIS**

In classical Twyman-Green or Fizeau type interferometry, the wavefront under test is compared with an ideal wavefront reflected from a reference plane or spherical surface. Systems based on these approaches are capable of yielding very high measuring accuracy, but are extremely sensitive to environmental disturbances such as vibration and turbulence. Moreover, they are characterized by a fixed sensitivity and thus cannot deal with strongly aberrated surfaces that result in high fringe densities.

Our FSIS system makes use of lateral shearing interferometry using a coarse **Ronchi grating** as the shearing device. In this approach, there is no need for a reference surface since the wavefront under test is made to interfere with a laterally shifted replica of itself. The wavefront under test is passed through the grating so that interference is obtained between the overlapping +1 and -1 diffraction orders produced by the grating. The result is an interferometer that is greatly insensitive to vibration and whose sensitivity can be varied by changing the pitch of the grating. However, the measured quantity is now the slope of the wavefront rather the wavefront itself as in classical interferometry and an additional numerical integration step is required to obtain the desired wavefront function.

Choosing a coordinate system where x and y are respectively in the directions of the short and long side of the cylindrical-type mirror, and orienting the grating with its pitch in the direction of the long side of the cylindrical-type mirror, the interferogram intensity is given by the following expression:

I(x,y) = A + B Cos [Phi(x, y + s/2) - Phi(x, y - s/2)] ,

where s is the shear value and Phi is the phase of the wavefront under test.

The measured quantity is thus equal to

Phi(x, y + s/2) - Phi(x, y - s/2) = s (dPhi/dy) ,

i.e. the slope of the wavefront in the y direction as desired. The wavefront Phi(x,y) is then obtained by numerical integration of the slope data

Phi(x,y) =   1/s [Phi(x, y + s/2) - Phi(x, y - s/2)] dy .

A measurement of a long cylindrical-type mirror is obtained by scanning the mirror with an full-width cylindrical wavefront over a finite number of overlapping subapertures. The full-surface wavefront is then obtained by numerical synthesis by least square fitting of the series of subaperture measurements. The result of the measurement is a collection of wavefront profiles or traces in the direction of the long side of the mirror.

3.0 THE OPTICAL INTERFEROMETER SUBSYSTEM

3.1 System Operation

A schematic diagram of the optical interferometer system is shown in Figure 1 and includes the following:

- A coherent laser source
- A spatial filter to 'clean up' and expand the beam, including a microscope objective mounted on an X-Y translation stage and focused onto a rotating diffuser plate
- A lens L1 to collimate the laser beam diverging from the spatial filter
- A lens L2 to focus the beam onto the grating shearing device
- A coarse Ronchi grating having a couple of lines per millimeter, mounted on an X-Y-Z translation stage and a rotation stage for focusing and alignment purposes
- A high quality, large collimating lens L to produces plane wave illumination
- A high quality cylindrical lens to produce the desired cylindrical wavefront used to illuminate, in a null configuration, the cylindrical type surface under test at normal incidence as shown in top and side views
- A beam splitter plate that recombines the reflected beam onto the CCD array detector through some imaging optics
- A High resolution CCD camera to capture the interferograms.

The cylindrical mirror under test is mounted onto a long travel linear motorized stage. The stage is translated in discrete steps so that the mirror is fully covered by a number of overlapping subapertures as shown in Figure 2.

The Y-axis of the grating translation stage is equipped with a micro-stepper stage that permits to translate the grating in its plane to vary the phase of the interferogram by small fractional increments.

Both motorized stages, as well as the CCD camera are under computer control so that the operations of scanning and data acquisition are implemented in a fully automated fashion.

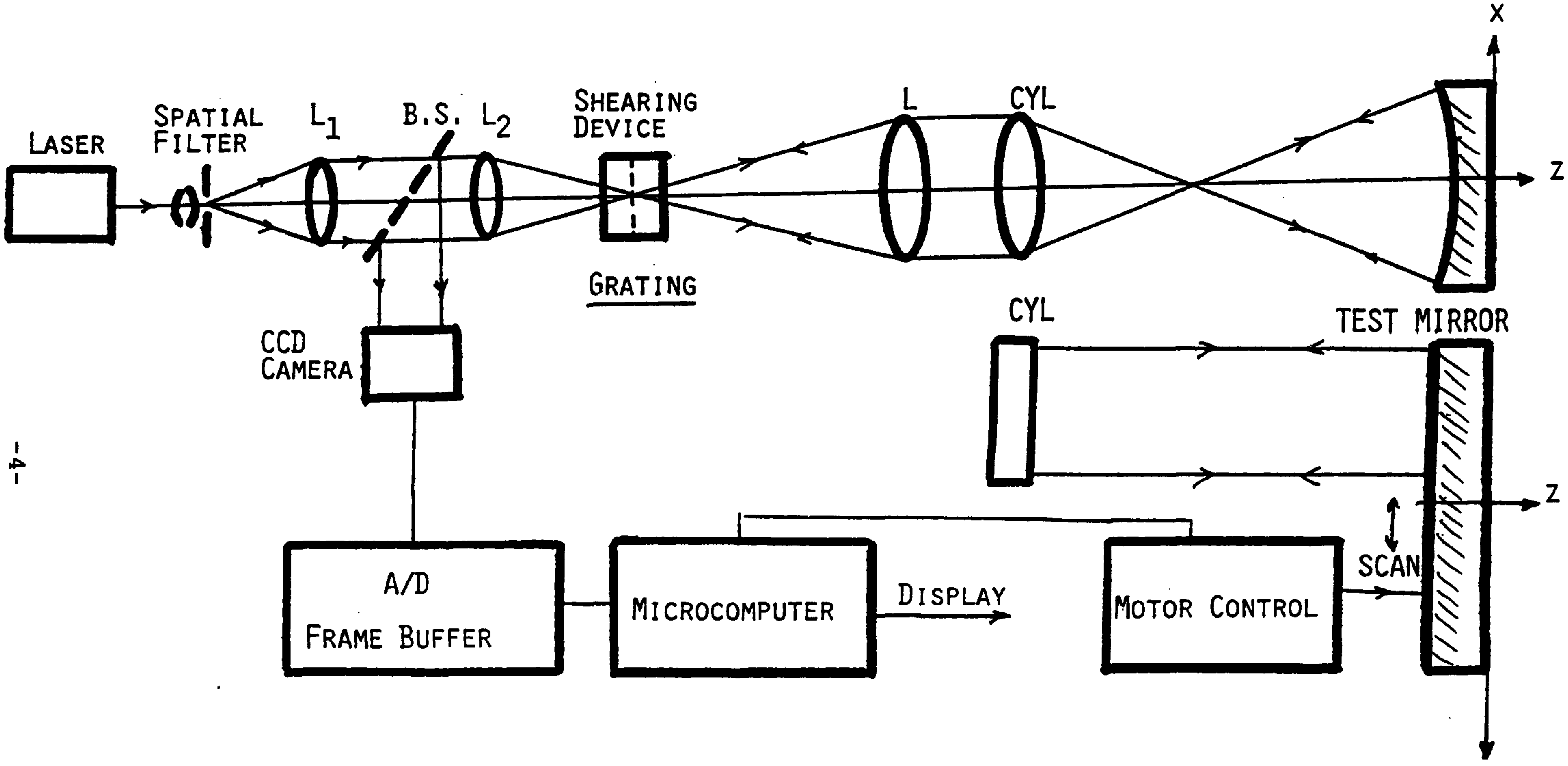


FIGURE 1: SCHEMATIC DIAGRAM OF INTERFEROMETER FOR FULL SURFACE TESTING OF CYLINDRICAL TYPE SURFACES

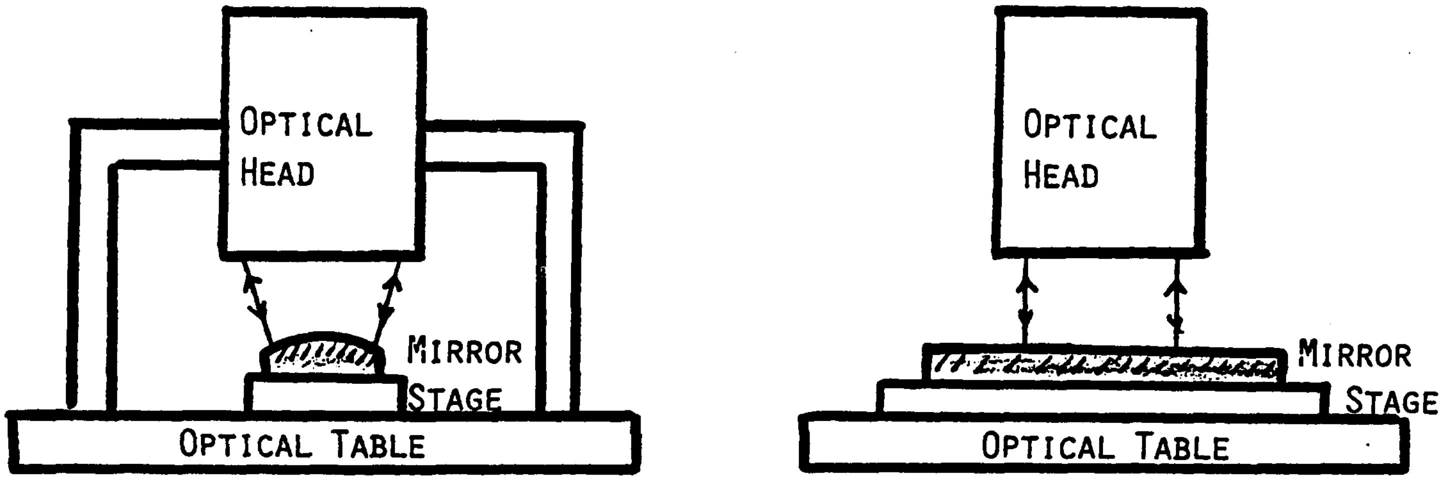


FIGURE 2: SCHEMATIC DIAGRAM OF THE COMPLETE SYSTEM

3.2 The Optical Prototype

Based on the design of Figure 1, an integrated prototype system was designed and constructed.

A diagram of the prototype system is shown in Figure 3, including:

- The laser source: 2mW helium-neon laser operating at 632.8nm
- The spatial filter with high power objective and rotating diffuser
- Beam shaping optics: collimating lens L1 and focusing lens L2 assembly
- Beam splitter plate
- Grating shearing device
- Reference beam forming optics for null testing: Collimating lens L and cylindrical lens CL assembly
- Imaging lens L3
- CCD camera detector.

Figures 4 to 7 show photographs of the prototype system:

- Figure 4 shows a photograph of the complete optical head and computer control
- Figure 5 shows the microstepper stage (ORIEL) which controls the grating position
- Figure 6 shows the spatial filter assembly with the rotating diffuser unit
- Figure 7 shows a close-up photograph of the long travel translation stage (NEAT) with the cylindrical mirror under test mounted via a tilt/rotation stage for alignment purposes.

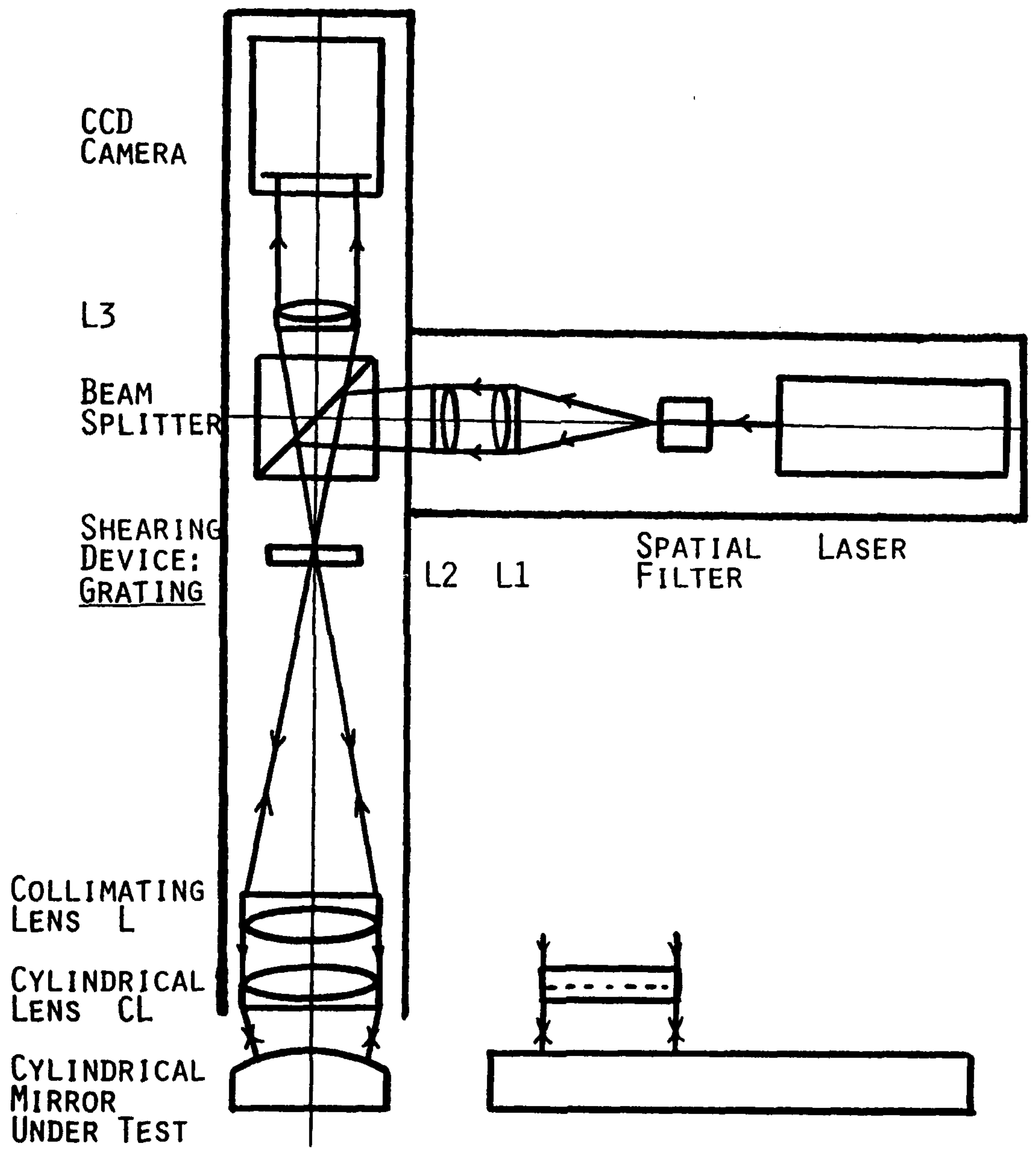


FIGURE 3: SCHEMATIC DIAGRAM OF INTERFEROMETRIC HEAD

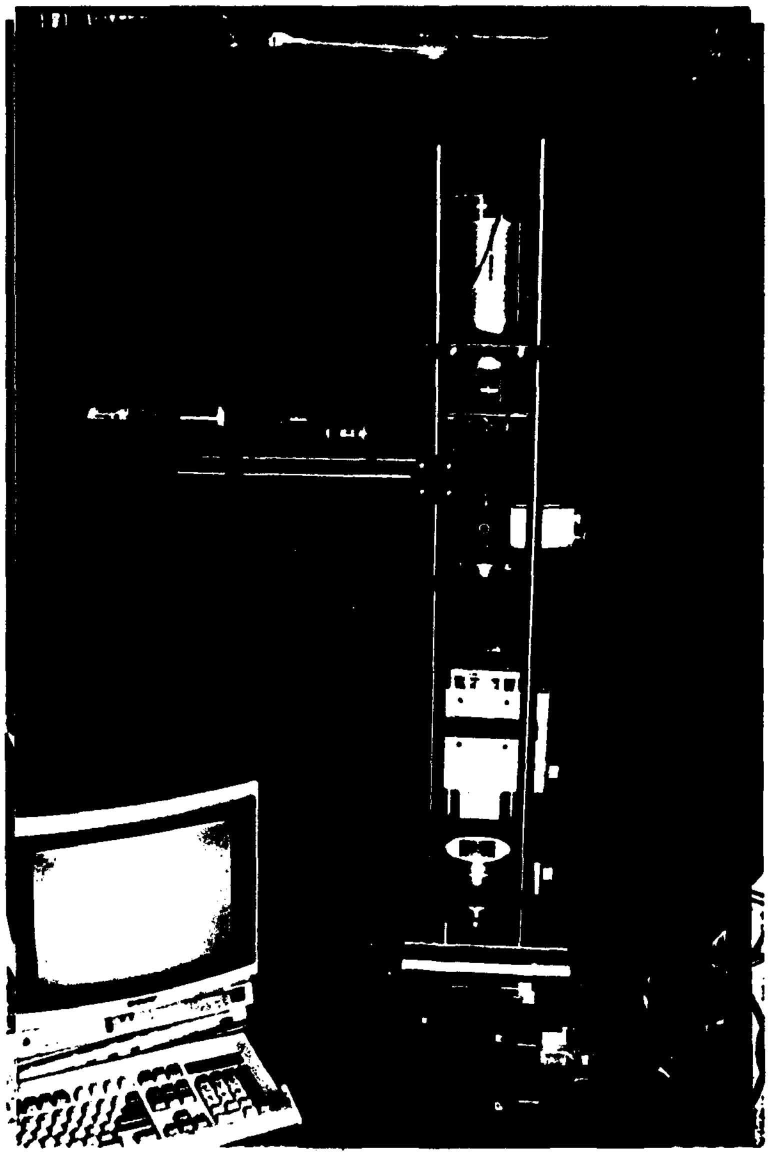


FIGURE 4: PHOTOGRAPH OF COMPLETE FSIS PROTOTYPE SYSTEM

FIGURE 5: PHOTOGRAPH OF FSIS SYSTEM SHOWING THE ORIEL STEPPER

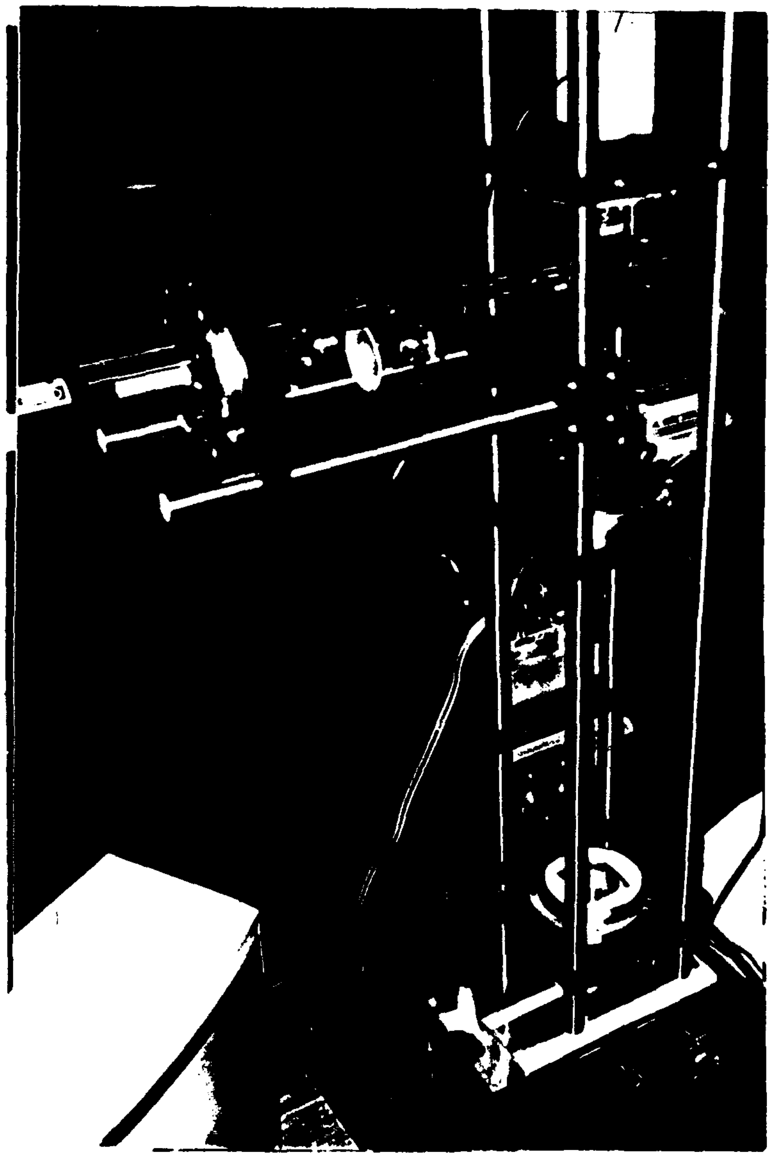


FIGURE 6: PHOTOGRAPH OF FSIS SYSTEM SHOWING THE ROTATING DIFFUSER

FIGURE 7: PHOTOGRAPH OF THE FSIS SYSTEM SHOWING THE TEST MIRROR ASSEMBLY

4.0 <u>DIGITAL AND ELECTRONICS HARDWARE</u>

The digital and electronics hardware for the FSIS includes

- The control electronics hardware: motor controllers to drive the test mirror scanning stage and the grating microstepper stage
- The data acquisition hardware: CCD camera and frame buffer to acquire, digitize and store the interferograms
- The data processing and display hardware: microcomputer system to process the interferograms to yield surface data and display the results.

Diagram 1 shows a schematic of the complete digital and electronics hardware used, including:

- Micrɔcomputer
- Hard and floppy disk drives
- VGA monitor
- Keyboard and mouse
- Laser printer
- Video printer
- Motor controllers (NEAT and ORIEL)
- Frame grabber board
- CCD camera.

BLOCK DIAGRAM OF THE DIGITAL/ELECTRONICS SYSTEM

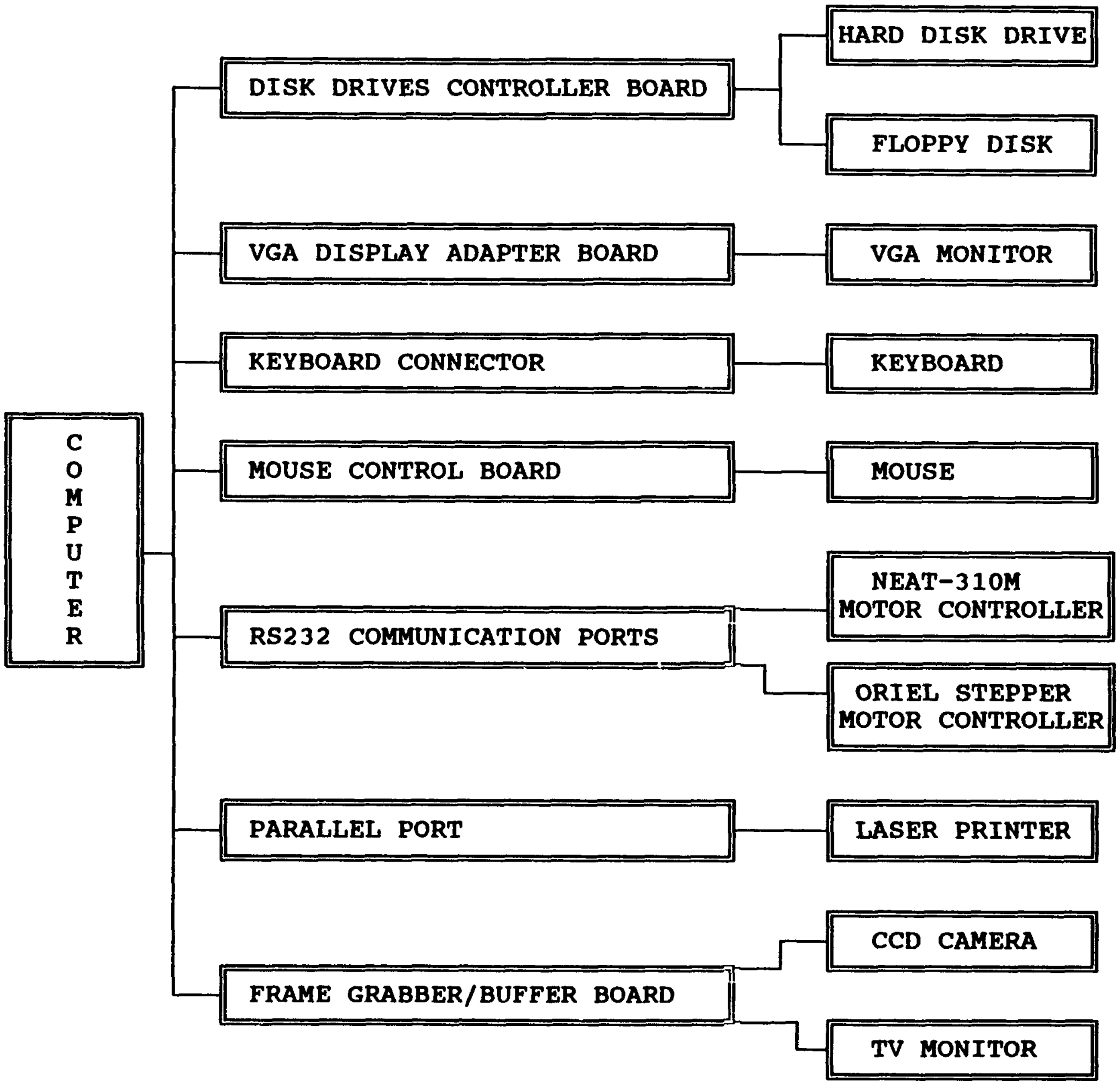

COMPUTER
DISK DRIVES CONTROLLER BOARD
HARD DISK DRIVE
FLOPPY DISK
VGA DISPLAY ADAPTER BOARD
VGA MONITOR
KEYBOARD CONNECTOR
KEYBOARD
MOUSE CONTROL BOARD
MOUSE
RS232 COMMUNICATION PORTS
NEAT-310M MOTOR CONTROLLER
ORIEL STEPPER MOTOR CONTROLLER
PARALLEL PORT
LASER PRINTER
FRAME GRABBER/BUFFER BOARD
CCD CAMERA
TV MONITOR

Diagram 1: Block Diagram of the Digital/Electronics System

4.1 Control Electronics Hardware

The control electronics hardware includes the motor controllers to drive

- The long travel scanning stage (NEAT) that holds and translate the mirror under test

- The short travel microstepper (ORIEL) that shifts the grating shearing device by fine incremental steps.

4.1.1 Scanning Stage Control (NEAT)

The NEAT translation stage is a long travel (24 inches) motorized stage under computer control as shown in Diagram 2 below.

The detailed specifications of the controller and the wiring connections with the computer are described in the following section entitled 'NEAT Translation Stage Control System'.

4.1.2 Microstepper Stage Control System (ORIEL)

The Oriel Microstepper stage is used to shift the grating shearing device within the interferometer head. It is controlled through the computer serial port RS232 COM1, as shown in Diagram 3.

The detailed specifications of the controller and the wiring connections with the computer are described in the following section entitled 'ORIEL Microstepper Control System'.

BLOCK DIAGRAM OF THE NEAT SCANNING STAGE CONTROL

STEPPING MOTOR CONTROLLER SYSTEM

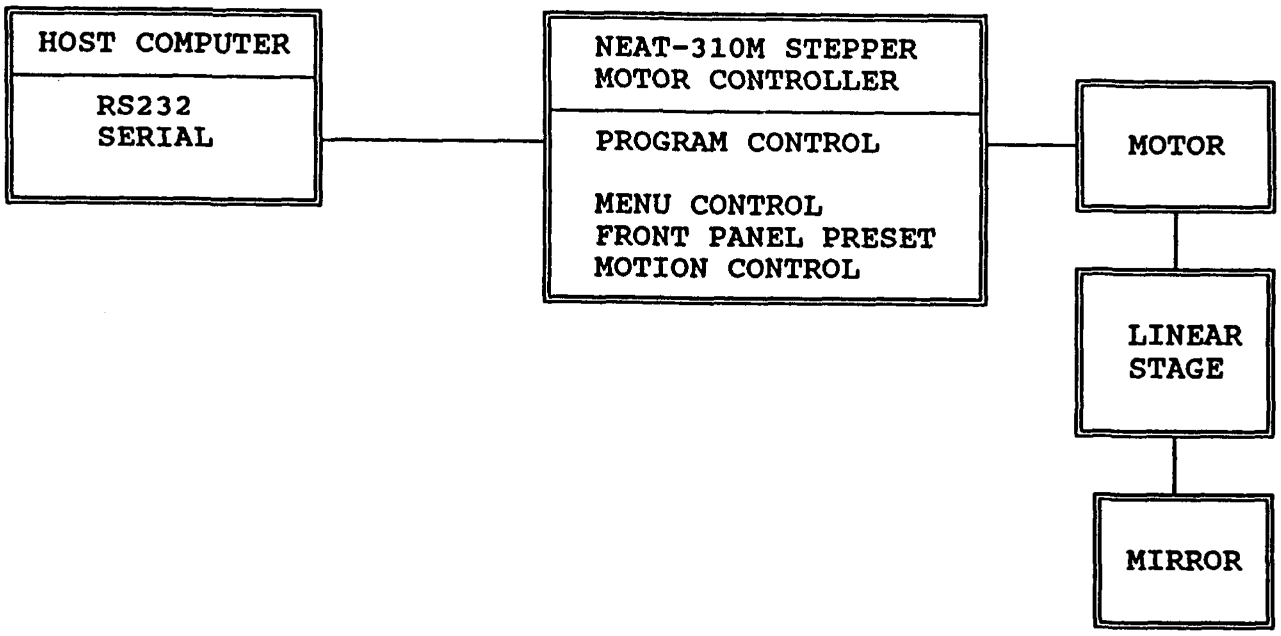


Diagram 2. Block Diagram of the NEAT Scanning Stage Control System

NEAT TRANSLATION STAGE CONTROL SYSTEM

1. NEAT-310M Controller Electronic/Mechanical Specification

- Drive Type:

  54 volt bipolar chopper
  .7 to 3.5 Amps/Phase, half-coil
  1.5 to 7 Amps/Phase, full-coil.

- Memory buffer:

  32 K bytes non-volatile program memory

- Velocity Range:

  40 to 327,640 steps / sec.

- Position Range:

  -8,388,608 to +8,388,608 steps

- Computer interface:

  RS 232 and 8-bit parallel communication ports

- Power requirements:

  115/230 V A.C. 50/60 Hz, 1.5 Amps

2. NEAT-310M Controller Communication Wiring Format

- Computer: RS-232 9 pin (Female) adapter.

  Pin #2: (Red)
  Pin #3: (Green)
  Pin #5: (Black)
  Pin #6,7,8: Short circuit.

- Controller: 25 pin (Male) adapter.

  Pin #2: (Green)
  Pin #3: (Red
  Pin #1,7: (Black)

3. **NEAT-310M Software Command Sets**

NEAT-310M controller uses RS-232 serial communications port to handle control commands from the PC computer. The communication port setting are: 8 data bit, 1 stop bit, no parity bit, ASCII data format.

The NEAT-310M operates with a command set of over 60 high-level ASCII commands. The following examples are common command sets we are using in the control software package:

- MR: Move relative.
  Format: MRddddd   ddddd range from -8,388,608 to + 8,388,608

- MA: Move absolute.
  Format: MAddddd   ddddd range from -8,388,608 to + 8,388,608

- MH: Move physical origin position.

- MC: Move continuous.
  Format MC+/-.

- SP: Set absolute register counter value.

- VI: Set initial velocity.*
  Format: VIddddd   ddddd range from 40 to 327,680
  default is 4000 steps / sec.

- VF: Set final velocity.
  Format: VFddddddd   ddddddd range from 40 to 327,680

- AC: Set acceleration and deceleration.
  Format: AC(+/-)ddddddd , '+' for acceleration, '-' for deceleration, without sign for both.

- ST: Stop moving immediately.

- ME: Mnemonic Expansion.
  Format: MFE for enable, MED for disable.

- MF: Move finish signal feed back.
  Format: MFE for enable, MFD for disable.

* Note: The primary resonance is at 400-1200 steps / sec. the maximum allowable start/stop frequency is load dependent, typically below 6000 / sec is safe.

BLOCK DIAGRAM OF THE ORIEL MICROSTEPPER CONTROL

STEPPING MOTOR CONTROLLER SYSTEM

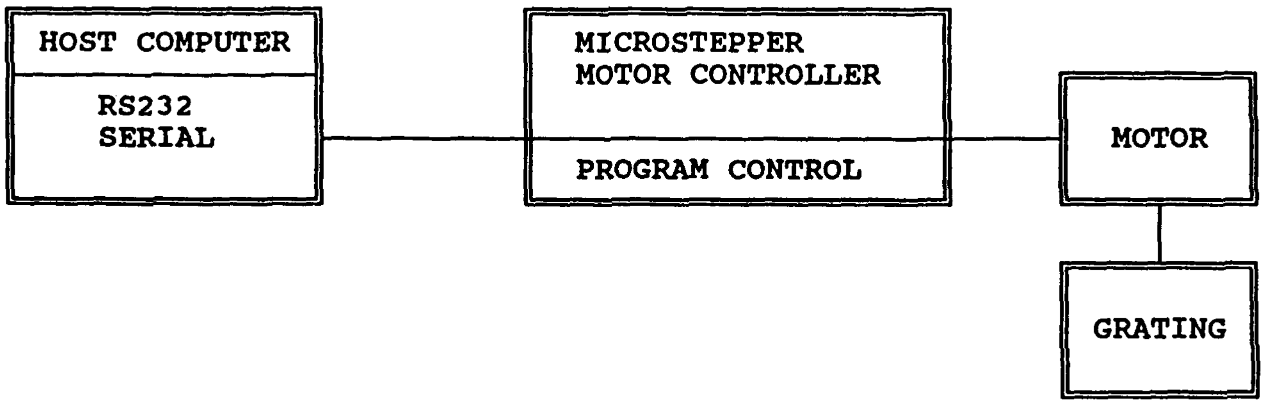


Diagram 3. Block Diagram of ORIEL Microstepper Control System

ORIEL MICROSTEPPER STAGE CONTROL SYSTEM

1. **Controller Electronic/Mechanical Specification:**

- Number of controlled stepper motors:

  Up to two, individually.

- Data I/O :

  300,600,1200,2400,4800,9600 baud, switch selectable.

- Compatible stepper motor:

  2 or 4 phase unipolar motor, 24V, 0.5 A max, per phase.

- Winding holding current:

  Reduced to 0.1 A per phase.

- Limit switch connections:

  Low logic level = limit reached.

- Maximum speed:

  Half step mode: 1000 steps / sec.
  Full step mode: 500 steps / sec.

- Computer interface:

  RS- 232 serial communication port

- Power requirements:

  110/220-240 VAC 50/60 Hz, switch selectable.

2. **Stepper Mike Motor**

Model: ORIEL Stepper Mike. Model 18500.

Specification:

- Step Size:

  Half step mode: 1.3, 0.7 um.
  Full step mode: 2 um.

- Maximum step rate:

  Half step mode: 1000 steps / sec.
  Full step mode: 500 steps / sec.

- Maximum spindle speed:

  1 mm /sec.

- Maximum axial load:

  15.5 lbs.

- Uni-directional repeatability:

  < 2 um.

- Backlash:

  < 3 um.

- Travel Range:

  0.5", (13 mm).

3. **Oriel Stepper Mike Controller Communication Port Wiring Format**

- Computer: RS232 9 pin (F) adapter.

  Pin #2: (Red)
  Pin #3: (Green)
  Pin #5: (Black)
  Pin #6,7,8: Short circuit.

- Controller: 25 pin (M) adapter.

  Pin #2: (Red)
  Pin #3: (Green)
  Pin #1,7: (Black)

- Controller to Motor:

  10 lines Rainbow cable.

4. **Oriel Stepper Mike Controller Software Command Sets**

Oriel Stepper Mike Controller uses RS-232 serial communications port to handle control commands from the PC computer. The communication port setting are: 7 data bit, 1 stop bit, no parity bit, ASCII data format.

The controller operates with command sets simular to NEAT-310M. The following list are some of the common command sets we will use in the control oftware package.

- A,B: Direct the flow of command, data, status inquiries to motor control register.

- C: Clear absolute register to zero, (set logic origin).

- D: Disable motor driver.

- E: Enable the motor driver.

- F: Change to full step mode.

- G: Go absolute.
  Format G +/- dddd, ',' is necessary to stop data field. dddd is the number of steps to go.

- H: Change to half step mode.

- I: Inquire the status of motor. return 'd', d = 0 - 7
  bit 0, direction: CW = 1 , CCW = 0.
  bit 1, step size: half = 1 , full = 0.
  bit 2, E/D status: Enable = 1, disable = 0.

- Q: Inquire the status of translator. return 'd', d = 0 - 7
  ```
  bit 0, motor:      On = 1 , Off = 0.
  bit 1, CW limit:   yes = 1 , no = 0.
  bit 2, CCW limit:  yes = 1 , no = 0.
  ```

- R: Set up the desired step rate for each motor.
  Format: Rn, n = 0 - 7.

- S: To start the moving.

- T: Travel relative steps.
  Format: Tddddddd,

- Z: Move to the logic origin.(the position of absolute register= 0)

- @: Unconditional stop both motor.

- >: Move single step in CW direction.

- <: Move single step in CCW direction.

- =: Start both motors simultaneously.

- * After command set, eg AT1000,BT1000.

- +: Set direction to CW.

- -: Set direction to CCW.

4.2 **Data Acquisition Hardware**

The data acquisition hardware includes:

- A high-resolution imaging CCD camera for interferogram detection
- A frame grabber and buffer for A/D conversion and storage
- A TV monitor for real-time interferogram viewing and a video printer for obtaining interferogram hard copying.

A block diagram of the complete data acquisition system is shown in Diagram 4.

4.2.1 The CCD Camera

The CCD camera used for interferogram data capture is an area camera with RS-170 video signal output made by COHU, INC.

The specifications of the camera are as follows:

- Imager:	Single CCD using the frame transfer method
- Image Area:	8.8mm x 6.6mm or 2/3 format
- Active Pixels:	754 x 484
- Resolution:	565 horizontal x 350 vertical TV lines
- Sensitivity:	0.07 lux useable picture, 0.16 lux at full video
- Signal/Noise:	56 dB
- Contrast Variation:	Less than 5% overall @ 25 degree C
- Gamma:	Jumper selectable at 1 or o.45
- AGC:	Jumper selectable, on/off
- Signal:	EIA RS-170 standard.

4.2.2 <u>The Frame Grabber/Buffer Board</u>

The frame grabber/buffer board used as an interface for the CCD camera is the Silicon Video Mux made by EPIX, INC.. The Silicon Video Mux is a single board frame grabber that allows the PC computer to digitize, process, display, transmit and archive video information.

The board can digitize one or several frames from a video camera such as a CCD camera and allow the PC to process the image data, and display the image data on a black/white or RGB monitor. Video data is digitized and displayed at 8 bits per pixel, and the board can contain up to 4 MB of image memory for multiple image storage.

The specifications of the frame buffer board are as follows:

- Variable sampling
- 1 MB or 4MB image memory
- Video memory address registers
- 752 pixels by 480 pixels maximum resolution
- External pixel clock, horizontal and vertical drive
- Multiple image storage
- Video line counters for selection of raster lines to be digitized/displayed.

4.2.3 <u>Output Devices</u>

The interferograms are viewed in real time on a TV monitor connected directly to the CCD camera or through the frame buffer.

A color or black/white video printer, connected to the monitor, is used to obtain hard copies of the interferograms.

A video recorder, also be connected to the monitor, can be used to record sequences of interferogram data.

CONNECTION DIAGRAM OF VIDEO INPUT/OUTPUT SYSTEM

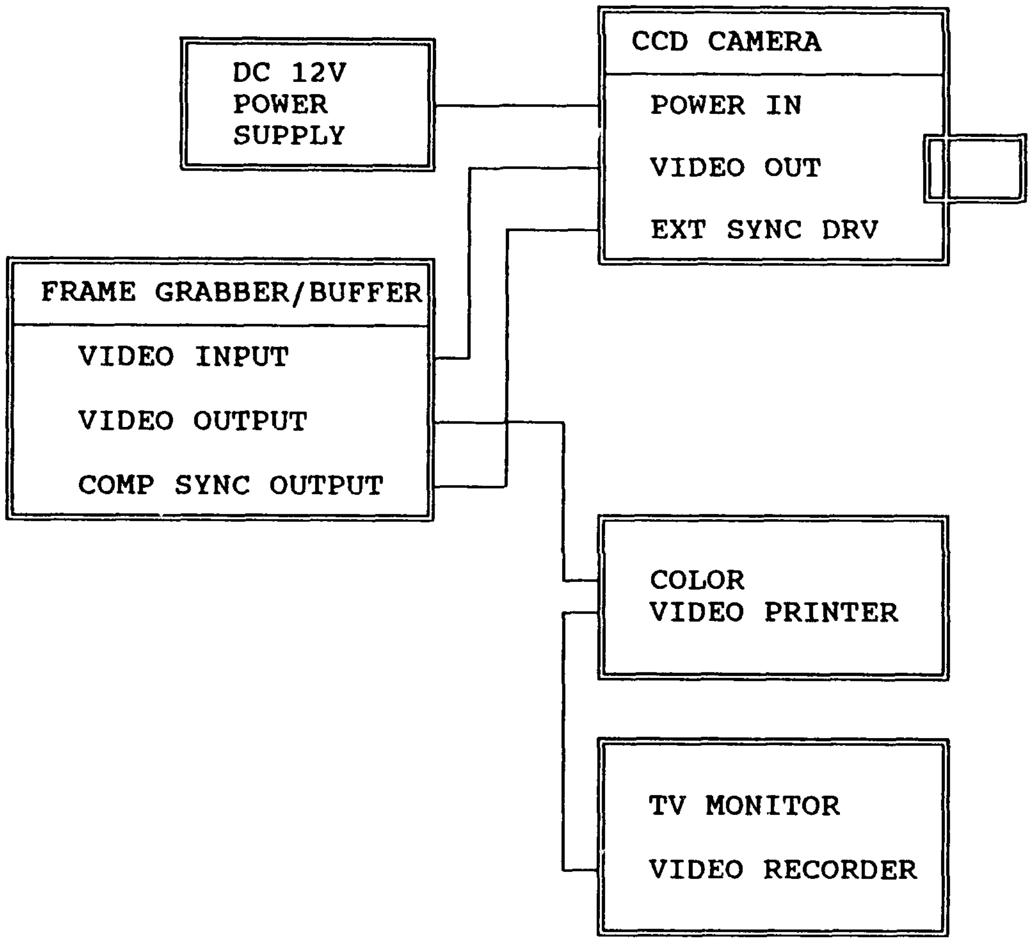


Diagram 4. Block diagram of the video input/output system

4.3 Data Processing Hardware

An IBM compatible microcomputer system is used for interferogram data processing.

The system includes:

- Computer with
  - 80486/40 MHs Microprocessor
  - 64K Cache Memory
  - 8 MB RAM
- Hard Disk Drive: 200 MB
- Dual Floppy Drives:
  - 5.25" Drive / 1.2 MB
  - 3.5" Drive / 1.44 MB
- Ports:
  - 1 Parallel Port
  - 2 Serial Ports
- Display Card:
  - VGA
  - Memory: 1 MB
- Monitor:
  - VGA Color
  - 0.28mm Non-Interlaced with 1024x768 Resolution
- 101 Enhanced Keyboard
- Microsoft Mouse
- MS-DOS 5
- Laser Printer: HP LaserJet III.

5.0 SOFTWARE

5.1 SOFTWARE OVERVIEW

The Multitrace system software package includes the following files:

0. Installation program: It installs the package software into the IBM PC compatible computer.

   PROGRAM: INSTALL.BAT

1. Batch file for compiling main program: It allows the user to modify the main program package and recompile into the executable file.

   PROGRAM: COMPILE1.BAT

2. Main program of the package: This program controls the Multitrace Profiler system, acquires the test object data, process and display the results.

   PROGRAM: PACKAGE.BAS
   PROGRAM: PACKAGE.EXE

3. Batch file for compiling demo program: It allows the user to modify the demo program package and recompile into the executable file.

   PROGRAM: COMPILE2.BAT

4. Demo program of the package: This demo program controls the Multitrace Profiler system, process the pre-stored the test object data and display the results.

   PROGRAM: DEMO.BAS
   PROGRAM: DEMO.EXE

5. Data set for demo purpose.

   PROGRAM: DEMOI.DAT
   PROGRAM: DEMOR.DAT

6. Image acquisition subroutine. (called by main program)

   PROGRAM: ACQ.C
   PROGRAM: ACQ.EXE

7. Compiler support libraries and subroutines.

   PROGRAM: BRUN30.LIB
   PROGRAM: USERLIB.EXE

5.2 User Manual: Operation of the Multi-trace System

The cylindrical mirror testing package user manual includes the hardware motion controllers specification sheets with sample programs to test the motion control and system operation menu driven software package with the programs.

The motion control specifications list the model and maker of the controllers and motors, their electrical and mechanical specifications, wire connections of the communication ports and control software command lists.

The system operation package is menu driven type of functions. When type "MAINF" in DOS system, the display will show a message and the MAIN MENU on the screen. By pressing the Function keys for the desired operation, the user can manipulate the interferometer system and implement some specific control and data processing for the test mirrors. There are two operations in the main menu and are implemented by pressing function keys F1, F2, pressing the "ESC" key will exit the program and back to DOS system.

The MAIN MENU is shown in the following:

```
     -   FSIS SYSTEM    -


     F1 : TEST MIRROR SCANNING

     F2 : FULL SURFACE MEASUREMENT

     ESC: EXIT
```

The functions of each operation are explained in the following:

F1. TEST MIRROR SCANNING

The TEST MIRROR SCANNING is to operate the interferometer system in the scanning mode, which allows the user to align the cylindrical test mirror with the interferometer and scan the full surface of the test mirror to view the complete interferogram.

When Function key F1 is pressed, the linear translation stage (NEAT) will be activated to move the test mirror along X-direction (cylindrical mirror axis) by five subaperture steps back and forth at a certain speed to allow the user to view and simulate the image acquisition of the full interferogram of the test mirror. This processing can be repeated as many times as wish by pressing the F1 as needed.

F2. FULL SURFACE MEASUREMENT

The FULL SURFACE MEASUREMENT is to operate the interferometer system to scan the cylindrical test mirror in a multi-aperture mode. For the first test mirror, it is assigned five subaperture to cover the full mirror.

It takes the first subaperture interferograms and stores in the frame buffer then move the test mirror to the 2nd position, then takes the 2nd subaperture interferograms and stores in the frame buffer. The same procedure repeats until all five subaperture interferograms are acquired and stored in the frame buffer.

After all the subaperture interferograms being acquired. The computer starts to process the phase maps of each subaperture and synthesize these subaperture phase into full surface phase map. Then the screen display the 3-D plot of the synthesis slope phase map and allow the user to key in the section number on the surface and plot the section trace profile.

To escape from the interactive sectional profile screen, just type "999" follow the "ENTER" to move to next display "THIN LINE CONTOUR". Press "ENTER" to leave this page.

The same display and interactive procedure for synthesis wavefront of the test mirror.

ESC: MAIN MENU

When "ESC" key is pressed, the program will return back to MAIN MENU page.

5.3 SOFTWARE PACKAGE SOURCE CODE

```
rem ***********************
rem *   INSTALL.BAT       *
rem ***********************

@echo off
echo Caution !!!
echo To prevent from version conflict, this installation will
echo delete all pre exist files echo in the destination directory.
echo Ctrl-Break to stop installation ; Or
pause
if '%1'=='' goto nodir
c: > NUL
md %1 > NUL
cd %1
echo Delete pre-exist files
del %1\*.* < yes > NUL
echo installing...
copy b:\*.*  > NUL
goto done
:nodir
c: > NUL
md \multra > NUL
cd \multra
echo Delete pre-exist files
del *.* < yes  > NUL
echo installing...
copy b:\*.*  > NUL
:done
echo Installation is done.
echo on


rem ***********************
rem *    COMPILE1.BAT     *
rem ***********************

qb package /l /d /q ;
link package ;


rem ***********************
rem *    COMPILE2.BAT     *
rem ***********************

qb demo /l /d /q ;
link demo ;


' ******************************
' *       PACKAGE.BAS          *
' ******************************
```

```
'This is the complete program package for the multitrace system

COMMON SHARED selkey    'declare command variable
c                 o                 n                 s                 t
thold%=200,cutr%=10,cutc%=20,ton%=1,toff%=0,center%=120,size%=240
,initord%=0
const scalepi=.5729583,smth%=1
const shift% = 3
const stepr%=3, xscale=4.2,yscale=2.5,yshift=3.8,stpno% = 3
const xdc = 20,ydc = 170,lll% = 1,rrr% = 120
common shared top%,bot%,lb%,rb%
common shared max,min,buf%
common shared row%,set%
common shared coe1,coe2,coe3
common shared maxp,minp,maxi,mini,rmsp,rmsi,pvp,pvi,rms,pv
common shared level1%,zscale,deginc,mark%,title$
rem $dynamic
d     i     m           s     h     a     r     e     d
obj(240),c%(240),r%(240),zzz(120,5),tz(360),ttz(360),shiftpts%(5)
d     i     m           s     h     a     r     e     d
order%(240),slope(120,120),zz(120,120),xb%(2,120),bb%(2,5),jff%(5
),jee%(5),filenm$(5)
dim shared colis%(8)
dim shared m(480),mp(480)
d     i     m           s     h     a     r     e     d
z(150),h(9),ish(9),xh(5),yh(5),im(9),x(120),y(120),jm(5),castab(3
,3,3)
rem $static
d                       a                       t                       a
0,0,8,0,0,2,5,1,7,6,9,2,3,4,5,6,0,3,4,7,1,0,1,8,4,3,0,9,0,1,2,3,9
,6,7,4,5,2,0,5,8,0,0,6,7,8,9,1,3,5,0,3,8,0,0,3,8,0,3,6,9,0,1,1,0

screen 9

call printmenu
CALL initial
DO UNTIL selkey = 27                                    'ESC key to exit
program
   CALL kbhit                                      'wait for key hit from
operator
     SELECT CASE selkey
     CASE 59                                        'have the stage moved
to each
       CALL send("MR+18750")                        'aperture for testing
scanning
       call stopped
       CALL send("MR+18750")
       call stopped
       CALL send("MR+18750")
       call stopped
       CALL send("MR+18750")
       call stopped
       CALL send("MR-75000")
       call stopped
       call printmenu
     CASE 60
       CLS
       LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #1
     "
       shell("acq 5")                              'call acq.exe to get the
image
       call waiting
       CALL send("MR+18750")
       call stopped
       LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #2
     "
       shell("acq 10")
       call waiting
       CALL send("MR+18750")
       call stopped
       LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #3
     "
       shell("acq 15")
       call waiting
       CALL send("MR+18750")
       call stopped
       LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #4
     "
       shell("acq 20")
       call waiting
       CALL send("MR+18750")
       call stopped
       LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #5
     "
       shell("acq 2[illegible]
       call waiting
       CALL send("MR-,5000")
       call stopped
       call main
       call printmenu
     CASE ELSE
     END SELECT
   LOOP
cls
END
'**
'*** Dummy waiting subroutine.
'**
sub waiting static
    for ii = -32767 to 32767
     for jj = 0 to 1
         next jj
     next ii
```

```
end sub


'**
'*** main process, contains computation , and display.
'**
sub main static
call init
call onoff(toff%)
for set% = 1 to 5
    buf% = set% * 5
    max = -32767
    min = 32767
    locate 12,20
    print "          COMPUTING SUBAPERTURE";set%;"
        "
    call compute
    next set%
call onoff(ton%)
call dodata
call display
end sub


'**
'*** wait for asterisk to make sure command is accept by controller
'**
SUB ASTERISK STATIC
    DO UNTIL INPUT$(1, #6) = "*"
       LOOP
END SUB


'**
'*** initial process
'**
SUB initial STATIC

OPEN "COM2:9600,N,8,1,RS,CS,DS,CD" FOR RANDOM AS #6  'OPEN SERIAL
PORT
CALL send("MED")                              'disable mnemonics
CALL send("MFE")                              'enable move finish
CALL send("VI600")                           'set initial velocity
CALL send("VF6000")                    'set final velocity
CALL send("AC1001")                              'set acceleration and
deceleration
END SUB

'**
'*** accept key hit from keyboard
'**
SUB kbhit STATIC
   key$ = ""
   DO
   WHILE key$ = ""
      key$ = INKEY$
      WEND
   IF LEN(key$) > 1 THEN
      skey$ = MID$(key$, 2, 1)
     ELSE
      skey$ = MID$(key$, 1, 1)
      END IF
   selkey = ASC(skey$)
   LOOP UNTIL key$ <> ""
END SUB

'**
'*** subroutine print the operation menu onto screen
'**
SUB printmenu STATIC
CLS
LOCATE 3,1
print "                       CYLINDRICAL MIRROR MULTITRACE MEASUREMENT
"
print ""
print ""
print ""
print ""
print "                              F1 : TEST MIRROR SCANNING"
print ""
print "                              F2 : ACQUISITION AND MEASUREMENT"
print ""
print ""
print "                              ESC: EXIT"
END SUB

'**
'*** subroutine for sending command to controller
'**
SUB send (a$) STATIC
    PRINT #6, a$                                         'send command to
controller
    CALL ASTERISK                                    'wait until asterisk
is return
END SUB

'**
'*** wait until process finish
'**
SUB STOPPED STATIC
     DO UNTIL INPUT$(1, #6) = "F"
       LOOP
      for ii = -32767 to 32767
       for jj = 0 to 1
           next jj
       next ii
END SUB
```

```
'**
'*** initialize parameter.
'**
sub init static
    filenm$(1) = "r0.dat"
    filenm$(2) = "r1.dat"
    filenm$(3) = "r2.dat"
    filenm$(4) = "r3.dat"
    filenm$(5) = "r4.dat"
    colis%(1) = 10
    colis%(2) =  3
    colis%(3) =  9
    colis%(4) =  1
    colis%(5) =  4
    colis%(6) = 12
    colis%(7) = 14
    colis%(8) = 15
    im(0)=0 : im(1)=1 : im(2)=1 : im(3)=0
    jm(0)=0 : jm(1)=0 : jm(2)=1 : jm(3)=1
    for k=0 to 3
     for j=0 to 3
         for i=0 to 3
          read castab(k,j,i)
          next i
         next j
     next k
end sub

'**
'*** initialize parameter of slope display.
'**
sub initp static
    title$="CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE
FUNCTION"
    levell% = 12
    zscale = -.3
    deginc = 36
    mark% = 60
    for i% = 1 to 480
     m(i%) = 32767
     mp(i%) = 32767
      next i%
    max = maxp
    min = minp
    rms = rmsp
    pv = maxp - minp
end sub

'**
'*** initialize parameter of wavefront display.
'**
sub initi static
    title$="CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT
FUNCTION"
```

```
    levell% = 180
    zscale = -.005
    deginc = 720
    mark% = 1800
    for i% = 1 to 480
     m(i%) = 32767
     mp(i%) = 32767
     next i%
    max = maxi
    min = mini
    rms = rmsi
    pv = maxi - mini
end sub


'**
'*** compute slope of each aperture.
'**
sub compute static
    open filenm$(set%) for output as #7
    call baseorder
    if top% < cutr% * 2 then
       top% = cutr%
      else
       top% = top% / 2 + 1
       end if
    if bot% > (120 - cutr%) * 2 then
       bot% = 120 - cutr%
      else
       bot% = bot% / 2 - 1
       end if
    for i%=top% to bot%
     row% = i% * 2 + 1
     call getrowphase
     if lb% < cutc% * 2 then
        xb%(1,i%) = cutc%
       else
        xb%(1,i%) = lb% / 2 + 1
        end if
     if rb% > (120 - cutc%) * 2 then
        xb%(2,i%) = 120 - cutc%
       else
        xb%(2,i%) = rb% / 2 - 1
        end if
     for j% = xb%(1,i%) to xb%(2,i%)
         if j% * 2 > lb% and j% * 2 < rb% then
            if obj(j% * 2) <> -32767 then
            zz(j%,i%) = obj(j%*2)
           else
            zz(j%,i%) = -32767
            end if
           else
            zz(j%,i%) = -32767
            end if
            next j%
        next i%
       call smooth
       call rotate
       print #7,top%,bot%,max,min
       for i% = top% to bot%
        print #7,i%,xb%(1,i%),xb%(2,i%)
        for j% = xb%(1,i%) to xb%(2,i%)
            print #7,slope(j%,i%)
            next j%
        next i%
   close
   end sub

   '**
   '*** trace the order of phase.
   '**
   sub traceord static
       order%(center%)=initord%
       cc%=initord%
       for j%=center% to size%-1
        if obj(j%+1)-obj(j%)<-180.0 then cc%=cc%+1
        if obj(j%+1)-obj(j%)>180.0 then cc%=cc%-1
        order%(j%+1)=cc%
        next j%
       cc%=initord%
       for j%=center% to 1 step -1
        if j%<>center% then
           if obj(j%+1)-obj(j%)<-180.0 then cc%=cc%-1
           if obj(j%+1)-obj(j%)>180.0 then cc%=cc%+1
        end if
        order%(j%)=cc%
        next j%
   end sub
   '**
   '*** convert the phase.
   '**
   sub phaseconversion(t%(1)) static
       for j%=1 to 240
        if t%(j%) <> -32767 then
           obj(j%)=t%(j%)*scalepi / 2
          else
           obj(j%) = -32767
           end if
        next j%
   end sub

   '**
   '*** get the base order.
   '**
   sub baseorder static
       call ptr86(sega%, offa%, varptr(c%(1)))
                                          c        a        l        l
   getpc(top%,bot%,thold%,size%,size%,buf%,center%,sega%,offa%)
```

```
    call phaseconversion(c%())
    call traceord
end sub

'**
'*** get phase.
'**
sub getrowphase static
    call ptr86(sega%, offa%, varptr(r%(1)))
                                            c        a        l        l
getpir(lb%,rb%,thold%,order%(row%),size%,size%,buf%,row%,sega%,of
fa%)
    call phaseconversion(r%())
end sub

'**
'*** data smoothing.
'**
sub smooth static
    for i% = 1 to 120
     for j% = 1 to 120
         slope(j%,i%) = -32767
         next j%
     next i%
    for i% = top% to bot%
     for j%=xb%(1,i%) to xb%(2,i%)
         if zz(j%,i%) <> -32767 then
            sum = 0
            cnt = 0
            for ii% = i% - smth% to i% + smth%
             if ii% > top% and ii% <= bot% then
                for jj% = j% - smth% to j% + smth%
                 if jj% > xb%(1,ii%) and jj% < xb%(2,ii%) then
                   if zz(jj%,ii%) <> -32767 then
                   sum = sum + zz(jj%,ii%)
                   cnt = cnt + 1
                   end if
                   end if
                 next jj%
                end if
             next ii%
            slope(120-i%,j%) = sum / cnt
            if max < slope(120-i%,j%) then
            max = slope(120-i%,j%)
           else
            end if
            if min > slope(120-i%,j%) then
            min = slope(120-i%,j%)
           else
            end if
            end if
         next j%
     next i%
end sub
```

```
'**
'*** data rotating.
'**
sub rotate static
    ntop% = 32767
    nbot% = -32767
    for i% = top% to bot%
     if xb%(1,i%) < ntop% then ntop% = xb%(1,i%)
     if xb%(2,i%) > nbot% then nbot% = xb%(2,i%)
     next i%
    for i% = 1 to 120
     xb%(1,i%) = 121
     xb%(2,i%) = 0
     next i%
    for i% = ntop% to nbot%
     for j% = 1 to 120
         if slope(j%,i%) <> -32767 then
            xb%(1,i%) = j%
            j% = 121
            end if
         next j%
     for j% = 120 to 1 step -1
         if slope(j%,i%) <> -32767 then
            xb%(2,i%) = j%
            j% = 0
            end if
         next j%
     next i%
    for i% = ntop% to nbot%
     if xb%(1,i%) < xb%(2,i%) then
        top% = i%
        i% = 121
        end if
     next i%
    for i% = nbot% to ntop% step -1
     if xb%(1,i%) < xb%(2,i%) then
        bot% = i%
        i% = 0
        end if
     next i%
end sub

'**
'*** match data to a complete window.
'**
sub dodata static
for i% = 1 to 5
    shiftpts%(i%) = shift% * (i% - 1)
    next i%
max = -32767 : min = 32767
open "r0.dat" for input as #1
open "r1.dat" for input as #2
open "r2.dat" for input as #3
open "r3.dat" for input as #4
open "r4.dat" for input as #5

input #1,top1%,bot1%,max1,min1
input #2,top2%,bot2%,max2,min2
input #3,top3%,bot3%,max3,min3
input #4,top4%,bot4%,max4,min4
input #5,top5%,bot5%,max5,min5
top% = 0
if top1% > top% then top% = top1%
if top2% > top% then top% = top2%
if top3% > top% then top% = top3%
if top4% > top% then top% = top4%
if top5% > top% then top% = top5%
bot% = 121
if bot1% < bot% then bot% = bot1%
if bot2% < bot% then bot% = bot2%
if bot3% < bot% then bot% = bot3%
if bot4% < bot% then bot% = bot4%
if bot5% < bot% then bot% = bot5%
for i% = top1% to top% - 1
    input #1,dmy%,b1%,b2%
    for j% = b1% to b2%
     input #1,z%
     next j%
    next i%
for i% = top2% to top% - 1
    input #2,dmy%,b1%,b2%
    for j% = b1% to b2%
     input #2,z%
     next j%
    next i%
for i% = top3% to top% - 1
    input #3,dmy%,b1%,b2%
    for j% = b1% to b2%
     input #3,z%
     next j%
    next i%
for i% = top4% to top% - 1
    input #4,dmy%,b1%,b2%
    for j% = b1% to b2%
     input #4,z%
     next j%
    next i%
for i% = top5% to top% - 1
    input #5,dmy%,b1%,b2%
    for j% = b1% to b2%
     input #5,z%
     next j%
    next i%
for i% = top% to bot%
    input #1,dmy%,bb%(1,1),bb%(2,1)
    nrb% = 120 - cut
    for j% = bb%(1,1) to bb%(2,1)
     if j% > nrb% then
```

```
      input #1,dmy%
     else
      input #1,zzz(j%,1)
      end if
  next j%
 if bb%(2,1) > nrb% then bb%(2,1) = nrb%

 input #2,dmy%,bb%(1,2),bb%(2,2)
 for j% = bb%(1,2) + shiftpts%(2) to bb%(2,2) + shiftpts%(2)
  if j% < 1 or j% > 120 then
      input #2,dmy%
     else
      input #2,zzz(j%,2)
      end if
  next j%
 if bb%(1,2) < 1 + cut then
     bb%(1,2) = cut + 1 + shiftpts%(2)
    else
     bb%(1,2) = bb%(1,2) + shiftpts%(2)
     end if
 if bb%(2,2) > 120 - cut then
     bb%(2,2) = 120 - cut + shiftpts%(2)
    else
     bb%(2,2) = bb%(2,2) + shiftpts%(2)
     end if
 if bb%(2,2) > 120 then bb%(2,2) = 120


 input #3,dmy%,bb%(1,3),bb%(2,3)
 for j% = bb%(1,3) + shiftpts%(3) to bb%(2,3) + shiftpts%(3)
  if j% < 1 or j% > 120 then
      input #3,dmy%
     else
      input #3,zzz(j%,3)
      end if
  next j%
 if bb%(1,3) < 1 + cut then
     bb%(1,3) = cut + 1 + shiftpts%(3)
    else
     bb%(1,3) = bb%(1,3) + shiftpts%(3)
     end if
 if bb%(2,3) > 120 - cut then
     bb%(2,3) = 120 - cut + shiftpts%(3)
    else
     bb%(2,3) = bb%(2,3) + shiftpts%(3)
     end if
 if bb%(2,3) > 120 then bb%(2,3) = 120


 input #4,dmy%,bb%(1,4),bb%(2,4)
 for j% = bb%(1,4) + shiftpts%(4) to bb%(2,4) + shiftpts%(4)
  if j% < 1 or j% > 120 then
      input #4,dmy%
     else
     input #4,zzz(j%,4)
     end if
  next j%
 if bb%(1,4) < 1 + cut then
    bb%(1,4) = cut + 1 + shiftpts%(4)
   else
    bb%(1,4) = bb%(1,4) + shiftpts%(4)
    end if
 if bb%(2,4) > 120 - cut then
    bb%(2,4) = 120 - cut + shiftpts%(4)
   else
    bb%(2,4) = bb%(2,4) + shiftpts%(4)
    end if
 if bb%(2,4) > 120 then bb%(2,4) = 120


 input #5,dmy%,bb%(1,5),bb%(2,5)
 for j% = bb%(1,5) + shiftpts%(5) to bb%(2,5) + shiftpts%(5)
  if j% < 1 or j% > 120 then
     input #5,dmy%
    else
     input #5,zzz(j%,5)
     end if
  next j%
 if bb%(1,5) < 1 + cut then
    bb%(1,5) = cut + 1 + shiftpts%(5)
   else
    bb%(1,5) = bb%(1,5) + shiftpts%(5)
    end if
 if bb%(2,5) > 120 - cut then
    bb%(2,5) = 120 - cut + shiftpts%(5)
   else
    bb%(2,5) = bb%(2,5) + shiftpts%(5)
    end if
 if bb%(2,5) > 120 then bb%(2,5) = 120
 for ii% = 1 to 5
  for j% = 1 to bb%(1,ii%) - 1
      zzz(j%,ii%) = -32767
      next j%
  for j% = bb%(2,ii%) + 1 to 120
      zzz(j%,ii%) = -32767
      next j%
  next ii%

 for j% = 1 to 360
  tz(j%) = -32767
  next j%
 for ii% = 1 to 4
  ni% = ii% + 1
  sum = 0
  cnt = 0
  jff%(ii%) = bb%(1,ni%) + 60
  jee%(ii%) = bb%(2,ii%)
  for j% = jff%(ii%) to jee%(ii%)
```

```
          nj% = j% - 60
          if zzz(j%,ii%) <> -32767 and zzz(nj%,ni%) <> -32767 then
             sum = sum + (zzz(j%,ii%) - zzz(nj%,ni%))
             cnt = cnt + 1
             end if
          next j%
      if cnt > 0 then
         dev = sum / cnt
        else
         dev = 0
         end if
      for j% = bb%(1,ni%) to bb%(2,ni%)
          if zzz(j%,ni%) <> -32767 then zzz(j%,ni%) = zzz(j%,ni%) +
dev
          next j%
      pi% = ii% - 1
      jdc% = pi% * 60
      if ii% = 1 then
         for j% = bb%(1,ii%) to jff%(ii%)
             tz(j%) = zzz(j%,ii%)
             'print ii%,j%,tz(j%)
             next j%
        else
         for j% = jee%(pi%) - 60 to jff%(ii%)
             tz(j%+jdc%) = zzz(j%,ii%)
             'print ii%,j%+jdc%,tz(j%+jdc%)
             next j%
         end if
      for j% = jff%(ii%) to jee%(ii%)
          nj% = j% - 60
          if (zzz(j%,ii%) <> -32767 and zzz(nj%,ni%) <> -32767) then
             tz(j%+jdc%) = (zzz(j%,ii%) + zzz(nj%,ni%)) / 2
            else
             if zzz(j%,ii%) <> -32767 then
             tz(j%+jdc%) = zzz(j%,ii%)
            else
             if zzz(nj%,ni%) <> -32767 then
                tz(j%+jdc%) = zzz(nj%,ni%)
               else
                tz(j%+jdc%) = -32767
                end if
             end if
             end if
          next j%
      if ii% = 4 then
         for j% = jee%(ii%) - 60 to bb%(2,ni%)
             tz(j%+jdc%+60) = zzz(j%,ni%)
             'print ii%,j%+jdc%+60,tz(j%+jdc%+60)
             next j%
         end if
      next ii%
   zone% = 3
   smpts% = 5
   ptp% = 2
```

```
 ptn% = 3
 for ii% = 1 to 4
     sum = 0
     cnt = 0
     ff% = jff%(ii%) + 60 * (ii% - 1) - zone%
     ee% = jff%(ii%) + 60 * (ii% - 1) + zone%
     for j% = ff% - ptp% to ff% + ptn% - 1
      if tz(j%) <> -32767 then
         sum = sum + tz(j%)
         cnt = cnt + 1
         end if
      next j%
     for j% = ff% to ee%
      if tz(j%) <> -32767 then ttz(j%) = sum / cnt
      if tz(j% - ptp%) <> -32767 then
         sum = sum - tz(j% - ptp%)
         cnt = cnt - 1
         end if
      if tz(j% + ptn%) <> -32767 then
         sum = sum + tz(j% + ptn%)
         cnt = cnt + 1
         end if
      next j%
     for j% = ff% to ee%
      if tz(j%) <> -32767 then tz(j%) = ttz(j%)
      next j%
     sum = 0
     cnt = 0
     ff% = jee%(ii%) + 60 * (ii% - 1) - zone%
     ee% = jee%(ii%) + 60 * (ii% - 1) + zone%
     for j% = ff% - ptp% to ff% + ptn% - 1
      if tz(j%) <> -32767 then
         sum = sum + tz(j%)
         cnt = cnt + 1
         end if
      next j%
     for j% = ff% to ee%
      if tz(j%) <> -32767 then ttz(j%) = sum / cnt
      if tz(j% - ptp%) <> -32767 then
         sum = sum - tz(j% - ptp%)
         cnt = cnt - 1
         end if
      if tz(j% + ptn%) <> -32767 then
         sum = sum + tz(j% + ptn%)
         cnt = cnt + 1
         end if
      next j%
     for j% = ff% to ee%
      if tz(j%) <> -32767 then tz(j%) = ttz(j%)
      next j%
     next ii%
for j% = 1 to 120
     zz(j%,i%) = -32767
     next j%
```

```
    for j% = 4 to 117
        zz(j%,i%) = tz(j% * 3)
        if zz(j%,i%) <> -32767 then
        if max < zz(j%,i%) then max = zz(j%,i%)
        if min > zz(j%,i%) then min = zz(j%,i%)
        end if
        next j%
    xb%(1,i%) = 121
    xb%(2,i%) = 0
    for j% = 1 to 120
        if zz(j%,i%) <> -32767 then
        xb%(1,i%) = j%
        j% = 121
        end if
        next j%
    for j% = 120 to 1 step -1
        if zz(j%,i%) <> -32767 then
        xb%(2,i%) = j%
        j% = 0
        end if
        next j%
    next i%
call tiltterm
call removetilt
call integrate
end sub


'**
'*** integrate the wavefront.
'**
sub integrate static
     maxi = -32767
     mini = 32767
     sumi = 0 : cnti = 0
     for i% = 1 to 120
      for j% = 1 to 120
          zz(j%,i%) = -32767
          next j%
      next i%
     for i% = top% to bot%
      zz(60,i%) = 0
      for j% = 61 to xb%(2,i%)
          if slope(j%,i%) <> -32767 then
             zz(j%,i%) = zz(j%-1,i%)+slope(j%,i%) / 1.5
            else
             zz(j%,i%) = zz(j%-1,i%)
             end if
          if maxi < zz(j%,i%) then maxi = zz(j%,i%)
          if mini > zz(j%,i%) then mini = zz(j%,i%)
          sumi = sumi + zz(j%,i%)^2
          cnti = cnti + 1
      next j%
      for j% = 59 to xb%(1,i%) step - 1
```

```
          if slope(j%,i%) <> -32767 then
             zz(j%,i%) = zz(j%+1,i%)-slope(j%+1,i%) / 1.5
            else
             zz(j%,i%) = zz(j%+1,i%)
             end if
          if maxi < zz(j%,i%) then maxi = zz(j%,i%)
          if mini > zz(j%,i%) then mini = zz(j%,i%)
          sumi = sumi + zz(j%,i%)^2
          cnti = cnti + 1
      next j%
     next i%
     rmsi = sqr(sumi / cnti)

end sub

'**
'*** remove the tilt plane of wavefront
'**
sub removetilt static
    maxp = -32767
    minp = 32767
    sump = 0 : cntp = 0
    for i% = 1 to 120
     for j% = 1 to 120
         slope(j%,i%) = -32767
         next j%
     next i%
    for i% = top% to bot%
     for j% = xb%(1,i%) to xb%(2,i%)
         if zz(j%,i%) <> -32767 then
            slope(j%,i%)= zz(j%,i%)-(coe1+coe2*j%+coe3*i%)
            if maxp < slope(j%,i%) then maxp = slope(j%,i%)
            if minp > slope(j%,i%) then minp = slope(j%,i%)
            sump = sump + slope(j%,i%)^2
            cntp = cntp + 1
            end if
     next j%
    next i%
    rmsp = sqr(sump / cntp)

end sub

'**
'*** compute the coefficient of tilt plane
'**
sub tiltterm static
     n = 0
     x = 0
     y = 0
     xy = 0
     xx = 0
     yy = 0
     w = 0
     wx = 0
     wy = 0
    for i% = top% to bot%
     for j% = xb%(1,i%) to xb%(2,i%)
         if zz(j%,i%) <> -32767 then
            iii = i%
            jjj = j%
            zzz = zz(j%,i%)
            n=n+1
            x=x+jjj
            y=y+iii
            xx=xx+jjj^2
            yy=yy+iii^2
            xy=xy+iii*jjj
            w=w+zzz
            wx=wx+zzz*jjj
            wy=wy+zzz*iii
            end if
         next j%
     next i%

     det=N*xx*yy+x*xy*y+x*y*xy-xx*y*y-N*xy*xy-x*x*yy
     coe1=(w*xx*yy+wx*xy*y+x*xy*wy-xx*y*wy-w*xy*xy-wx*yy*x)/det
     coe2=(N*wx*yy+x*y*wy+w*xy*y-wx*y*y-N*wy*xy-w*x*yy)/det
     coe3=(N*xx*wy+x*w*xy+x*y*wx-y*w*xx-N*xy*wx-x*x*wy)/det
end sub


'**
'*** display the result.
'**
sub display static
    call initp
    call plot(slope())
    ys = 2 : xs = 4.5 : xsp =  (640 - xs * 120)/2 : ysp = 50 - top
* ys
    call contour(slope(),lll%,rrr%,top%,bot%,xs,ys,xsp,ysp)
    key$ = ""
    WHILE key$ = ""
       key$ = INKEY$
       WEND
    call initi
    call plot(zz())
    ys = 2 : xs = 4.5 : xsp =  (640 - xs * 120)/2 : ysp = 50 - top
* ys
    call contour(zz(),lll%,rrr%,top%,bot%,xs,ys,xsp,ysp)
    key$ = ""
    WHILE key$ = ""
       key$ = INKEY$
       WEND
end sub


'**
'*** hidden line remove routine.
```

```
'**
sub hiddenline(x1,x2,y1,y2,m1,m2,mm1,mm2,col%) static

       '-------  check the hidden line ----------------
      if y1<m1 then
         if y2<m2 then
            mm1=y1
            mm2=y2
            line (x1,y1)-(x2,y2),colis%(col%)
         else
            m=(y2-y1)/(x2-x1)
            b=y1-m*x1
            d=(y1-m1)*(x2-x1)/(m2-y2+y1-m1)
            x=x1+d
            y=m*x+b
            line (x1,y1)-(x,y),colis%(col%)
            mm1=y1
            mm2=m2
         end if
      elseif y2<m2 then
         m=(y1-y2)/(x1-x2)
         b=y1-m*x1
         d=(m1-y1)*(x2-x1)/(y2-m2+m1-y1)
         x=x1+d
         y=m*x+b
         line (x,y)-(x2,y2),colis%(col%)
         mm1=m1
         mm2=y2
      else
         mm1=m1
         mm2=m2
      end if
end sub

'**
'*** 3 Dimensional hidden line removed display
'**
sub plot(arr(2)) static
    cls
    county% = 0
    for i% = bot% + stepr% to top% step -stepr%
     prehigh = 0
     for j%=lll% to rrr% step 1
         if arr(j%,i%) <> -32767 then
            high = arr(j%,i%)-min
            y2=high*zscale-county%*yshift+ydc
           else
            y2=-county%*yshift+ydc
            high = 0
            end if
         if high <> 0 then
            coli% = high / levell%
            coli% = coli% mod 7 + 1
           else
```

```
                if prehigh <> 0 then
                coli% = prehigh / levell%
                coli% = coli% mod 7 + 1
               else
                coli% = 8
               end if
               end if
             x2=j%*xscale+xdc
             if county%=0 then
               mp(j0%)=y2
               if j% > lll% then line (x1,y1)-(x2,y2),colis%(coli%)
              else
               if j% > lll% then
               max1=m(j%-1)
               max2=m(j%)
               call hiddenline(x1,x2,y1,y2,max1,max2,maxp1,maxp2,coli%)
               mp(j%-1)=maxp1
               mp(j%)=maxp2
               end if
               end if
            x1=x2:y1=y2
            prehigh = high
            next j%
         county% = county% + 1
         next i%
        for j%=1 to 480
         m(j%)=mp(j%)
        next j%
        call drawmark
        call boxf
        call sec(arr())
    end sub

    '**
    '*** contour display
    '**
    sub contour (arrd(2),xf%,xe%,yf%,ye%,xs,ys,xsp,ysp) static
    cls
    for i = 1 to 120
        x(i) = xsp + (i - 1) * xs
        next i
    for i = 1 to 120
        y(i) = ysp + (i - 1) * ys
        next i

    if min <= 0 and max >= 0 then
       exlevel = 1
      else
       exlevel = 0
       end if

    locate 22,64
    print levell% chr$(248);"/ LEVEL"
    call box1
```

```
mainp:
for i = 0 to 150
    z(i) = min + i * levell%
    if z(i) > max then
       level = i
       i = 200
       end if
    next i

for j=ye% - stpno% to yf% step -stpno%
    if j<top% or j>=bot% then goto noneinbox1
    for i=xf% to xe% - stpno% step stpno%
     if i<xb%(1,j) or i>xb%(2,j) - 1 or i < xb%(1,j+stpno%) or i >
xb%(2,j+stpno%) - 1 then goto noneinbox
     if  arrd(i,j)  =  -32767  or  arrd(i+stpno%,j)  =  -32767  or
arrd(i,j+stpno%) = -32767 or arrd(i+stpno%,j+stpno%) = -32767 then
goto noneinbox
     locate 23,10
     if (arrd(i,j)<arrd(i,j+stpno%)) then
        dmin=arrd(i,j)
     else
        dmin=arrd(i,j+stpno%)
     end if
     if arrd(i+stpno%,j)<dmin then dmin=arrd(i+stpno%,j)
     if   arrd(i+stpno%,j+stpno%)<dmin   then
dmin=arrd(i+stpno%,j+stpno%)
     if arrd(i,j)>arrd(i,j+stpno%) then
        dmax=arrd(i,j)
     else
        dmax=arrd(i,j+stpno%)
     end if
     if arrd(i+stpno%,j)>dmax then dmax=arrd(i+stpno%,j)
     if   arrd(i+stpno%,j+stpno%)>dmax   then
dmax=arrd(i+stpno%,j+stpno%)
     if dmax<z(0) or dmin>z(level-1) then goto noneinbox
     for k=0 to level - 1
         if z(k)<dmin or z(k)>dmax then goto noneintri
         for m=4 to 0 step -1
          if m>0 then
             h(m)=arrd(i+stpno%*im(m-1),j+stpno%*jm(m-1))-z(k)
             xh(m)=x(i+stpno%*im(m-1))
             yh(m)=y(j+stpno%*jm(m-1))
          end if
          if m=0 then
             h(0)=(h(1)+h(2)+h(3)+h(4))/4
             xh(0)=(x(i)+x(i+stpno%))/2
             yh(0)=(y(j)+y(j+stpno%))/2
          end if
          if h(m)>0 then
             ish(m)=2
          elseif (h(m)<0) then
             ish(m)=0
          else
             ish(m)=1
        end if
       next m
       for m=1 to 4
        m1=m:m2=0:m3=m+1
        if m3=5 then m3=1
        casetype=cint(castab(ish(m1),ish(m2),ish(m3)))
          if casetype=0 then
          goto case0
          end if
        o   n   c   a   s   e   t   y   p   e   g   o   t   o
case1,case2,case3,case4,case5,case6,case7,case8,case9
        case1:
             x1=xh(m1):y1=yh(m1):x2=xh(m2):y2=yh(m2)
             goto drawit
        case2:
             x1=xh(m2):y1=yh(m2):x2=xh(m3):y2=yh(m3)
             goto drawit
        case3:
             x1=xh(m3):y1=yh(m3):x2=xh(m1):y2=yh(m1)
             goto drawit
        case4:
             x1=xh(m1):y1=yh(m1)
             x2=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
             y2=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
             goto drawit
        case5:
             x1=xh(m2):y1=yh(m2)
             x2=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
             y2=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
             goto drawit
        case6:
             x1=xh(m3):y1=yh(m3)
             x2=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
             y2=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
             goto drawit
        case7:
             x1=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
             y1=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
             x2=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
             y2=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
             goto drawit
        case8:
             x1=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
             y1=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
             x2=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
             y2=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
             goto drawit
        case9:
             x1=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
             y1=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
             x2=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
             y2=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
         drawit:
           x1 = int(x1)
```

```
              x2 = int(x2)
              y1 = int(y1)
              y2 = int(y2)
              line (x1,y1)-(x2,y2)
           case0: next m

noneintri: next k
noneinbox: next i
noneinbox1:  next j
10202 end sub


'**
'*** draw scale index
'**
sub drawmark static
    unitx = 535 : unity = 45
    unitsize = mark% * zscale

  line(unitx,unity)-(unitx,unity-unitsize)
  line(unitx-3,unity)-(unitx+3,unity)
  line(unitx-3,unity-unitsize)-(unitx+3,unity-unitsize)
  locate 4,70
  print mark%;chr$(248) ;
end sub



'**
'*** outline box
'**
sub boxf static
  locate 1,8
  print title$
  line(0,0)-(639,349),,b
  line(3,17)-(636,346),,b
  line(0,14)-(639,14)
  locate 14,50
  print "RMS";rms ;
  locate 14,65
  print "P-V";pv ;
end sub
'**
'*** outline box
'**
sub box1 static
  locate 1,8
  print title$
  line(0,0)-(639,349),,b
  line(3,17)-(636,346),,b
  line(0,14)-(639,14)
  locate 23,65
  print "RMS";rms ;
  locate 24,65
  print "P-V";pv  ;
end sub

'**
'*** sectional display routine
'**
sub sec(arr(2)) static
  yff = 290
  yee = 200
  mmff = 0
  xff = lll% * xscale + xdc
  xee = rrr% * xscale + xdc
  mmffxc = (xee - xff) / (rrr% - lll%) * 1.190625
  degsc = 85 / (max - min + deginc * .99)

  if top% mod 5 then
     ydfr = (top% / 5 + 1) * 5
    else
     ydfr = top%
     end if
  ydlr = bot% / 5 * 5

  for i% = ydfr to ydlr step 20
      xx = (rrr% + 5) * xscale + xdc
      yy = (i% - bot%) / stepr% * yshift + ydc
      line(xx,yy)-(xx+15,yy)
      row% = yy /14 + 1
      col% = xx / 8 + 3
      locate row%,col%
      print i% ;
      next i%
  secno% = 0
  do while secno% <> 999
     secno% = 0
     do until secno% = 999 or secno% >= top% and secno% <= bot%
     locate 23,5
     print "Key in trace no.";top%;"-";bot%; " 999 to quit " ;"
                                         " ;
     locate 23,50
     input secno%
     if secno% <> 999 and (secno% < top% or secno% > bot%) then
print chr$(7) ;
     loop
     if secno% <> 999 then
     call clearsec
     locate 23,35
     print "TRACE #";secno%
     locate 22,6
     print "(mm)" ;
     for degff=0 to (max-min) + deginc *.99 step deginc
         yy = -degff * degsc + yff
         line(xff,yy)-(xee,yy),,,&hcccc
         locate yy / 14 + 1,68
         print degff;chr$(248) ;
```

```
            next degff
        for mmff = 0 to 99 step 10
            xx = mmff * mmffxc + xff
            line(xx,yff)-(xx,yy),,,&hcccc
            mmm% = mmff
            if mmm% mod 20 = 0 then
               locate yff/14+1,xx/8
               print mmm% ;
               end if
            next mmff
        line(xee,yff)-(xee,yy) ,,,&hcccc

        first = 1
        sum = cnt = 0
        peak = -32767   : valley = 32767
        for j% = lll% to rrr%
            if arr(j%,secno%) <> -32767 then
               xx = j% * xscale + xdc
               yy = (min-arr(j%,secno%)) * degsc + yff
               if peak < arr(j%,secno%) then peak = arr(j%,secno%)
               if valley > arr(j%,secno%) then valley = arr(j%,secno%)
               sum = sum + arr(j%,secno%)^2
               cnt = cnt + 1
               if first <> 1 then line(xx,yy)-(xxo,yyo),14
               first = 0
               xxo = xx
               yyo = yy
               end if
            next j%
        locate 23,50
        print "RMS";sqr(sum/cnt) ;
        locate 23,65
        print "P-V";peak-valley ;
        key$ = ""
        WHILE key$ = ""
              key$ = INKEY$
              WEND
        end if
        loop
end sub

'**
'*** clean clipregin
'**
sub clearsec static
locate 15,2
print "
                 " ;
locate 16,2
print "
                 " ;
locate 17,2
print "
                 " ;
```

```
locate 18,2
print "
                              " ;
locate 19,2
print "
                              " ;
locate 20,2
print "
                              " ;
locate 21,2
print "
                              " ;
locate 22,2
print "
                              " ;
locate 23,2
print "
                              " ;
locate 24,2
print "
                              " ;
end sub


/* *************************
   *       ACQ.C           *
   *************************
*/

/* Get image by using EPIX and MDC3-BMCC */
#include <process.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <dos.h>
#include "svobj.h"              /* function prototypes */
#include "pxmp.h"
int xdim,ydim,images ;
long int buf,speed ;
char one[80],tot[80],tmp[80],spd[80] ;

main(msgn,msgc)
int msgn ;
char *msgc[] ;
{ long int bufs,bufe ;
  int i,ii,xwid,yintl,j,k,step,steps ;


  if (msgn < 2)
     { printf("\n 'Getimage XDIM YDIM BUFNO IMAGES STEPS SPEED' to
run ") ;
       printf("\nFunction : Get image by using EPIX and MDC3-BMCC")
;
```

```
       printf("\nParameter: ") ;
       printf("\n  Bufno   : Starting buffer no of images in SV
board") ;
       printf("\n  FILENM : IMAGE NAME to be save") ;
       exit(99) ;
       }

  xdim = ydim = 240 ;
  buf = atol(msgc[1]) ;
  step = 63 ;
  steps = step * 5 ;
  images = 5 ;
  ii = images - 1 ;
  sprintf(one,"+T%d,S",step) ;
  sprintf(tot,"-T%d,S",steps) ;


  opensi(240,3,0,240,-1,0) ;
  openoriel() ;
  bufs = buf ;
  bufe = bufs + 4 ;
     for (i = 0 ; i < 5 ; i++,buf++)
         {
          tocoml(one) ;
          for (j = -32767 ; j < 32767 ; j++)
           for (k = 0 ; k < 30 ; k++) ;
           pxd_video('z',buf) ;
           pxd_video('p',buf) ;
           }
  tocoml(tot) ;
/*  pxd_video('s',0L) ;
                                           i                 f
((errno=pxio8_fmbwrite(pxd_defimage(0L,0,0,-1,-1),bufs,bufe,msgc[
2],0L,1,1)) != 0)
     printf("\nError: %s",pxerrnomesg(errno)) ;   */
  pxd_close() ;
}


opensi(xdim,xwid,xoff,ydim,yoff,intla)
int xdim,xwid,xoff,ydim,yoff,intla ;
{ do_signal() ;
  if (pxd_svopen(0xD, 0x300, 5, 0xD)<0)
     { printf("Open Silicon Video Error \n");
       exit(99) ;
       }
  pxd_vidgenlock(1,1) ;
  pxd_vidparm(xdim,xwid,xoff,ydim,1,yoff,intla) ;
}
sigint()
{
 int abort;
   printf("\nInterrupt - closing SILICON VIDEO\n");
   pxd_close();
```

:

```
   exit(1);
}

sigfpe()
{
 int abort;
   printf("\nFloating point exception - closing SILICON VIDEO\n");
   pxd_close();
/*    _setvideomode(_TEXTC80) ;  */
   exit(1);
}

do_signal()
{
   signal(SIGINT, sigint);
   signal(SIGFPE, sigfpe);
}


openoriel()
{ int i;
  union REGS in,out ;
  char init[20] = "\x003\x00aA\0" ;

  (in).x.dx = 0 ;
  (in).h.al = 0xe2 ;
  (in).h.ah = 0 ;
  int86(0x14,&in,&out) ;
  tocom1("AEH") ;
}

tocom1(string)
char *string;
{
  int j,ch;
  for ( ; *string != '\0' ; string++)
      { for (j=-20000; j < 2000; j++);     /* the maximum speed of
1200 */
     ch = *string;
     outp (0x3f8,ch);
     }
}
```

```
' ***************************
' *        DEMO.BAS         *
' ***************************


'This is the complete demo package for the multitrace system

COMMON SHARED selkey    'declare command variable
c            o             n             s              t
thold%=200,cutr%=10,cutc%=20,ton%=1,toff%=0,center%=120,size%=240
,initord%=0
const scalepi=.5729583,smth%=1
const shift% = 3
const stepr%=3, xscale=4.2,yscale=2.5,yshift=3.8,stpno% = 3
const xdc = 20,ydc = 170,111% = 1,rrr% = 120
common shared top%,bot%,lb%,rb%
common shared max,min,buf%
common shared row%,set%
common shared coe1,coe2,coe3
common shared maxp,minp,maxi,mini,rmsp,rmsi,pvp,pvi,rms,pv
common shared level1%,zscale,deginc,mark%,title$
rem $dynamic
d   i    m          s    h    a    r    e    d
obj(240),c%(240),r%(240),zzz(120,5),tz(360),ttz(360),shiftpts%(5)
d   i    m          s    h    a    r    e    d
order%(240),slope(120,120),zz(120,120),xb%(2,120),bb%(2,5),jff%(5
),jee%(5),filenm$(5)
dim shared colis%(8)
dim shared m(480),mp(480)
d   i    m          s    h    a    r    e    d
z(150),h(9),ish(9),xh(5),yh(5),im(9),x(120),y(120),jm(5),castab(3
,3,3)
rem $static
d                   a                  t                  a
0,0,8,0,0,2,5,1,7,6,9,2,3,4,5,6,0,3,4,7,1,0,1,8,4,3,0,9,0,1,2,3,9
,6,7,4,5,2,0,5,8,0,0,6,7,8,9,1,3,5,0,3,8,0,0,3,8,0,3,6,9,0,1,1,0

screen 9

call printmenu
CALL initial
DO UNTIL selkey = 27                                 'ESC key to exit
program
   CALL kbhit                                   'wait for key hit from
operator
     SELECT CASE selkey
     CASE 59                                    'have the stage moved
to each
       CALL send("MR+18750")                     'aperture for testing
scanning
       call stopped
       CALL send("MR+18750")
       call stopped
       CALL send("MR+18750")
```

```
' **************************
' *       DEMO.BAS         *
' **************************


'This is the complete demo package for the multitrace system

COMMON SHARED selkey    'declare command variable
c               o               n               s               t
thold%=200,cutr%=10,cutc%=20,ton%=1,toff%=0,center%=120,size%=240
,initord%=0
const scalepi=.5729583,smth%=1
const shift% = 3
const stepr%=3, xscale=4.2,yscale=2.5,yshift=3.8,stpno% = 3
const xdc = 20,ydc = 170,lll% = 1,rrr% = 120
common shared top%,bot%,lb%,rb%
common shared max,min,buf%
common shared row%,set%
common shared coe1,coe2,coe3
common shared maxp,minp,maxi,mini,rmsp,rmsi,pvp,pvi,rms,pv
common shared levell%,zscale,deginc,mark%,title$
rem $dynamic
d   i     m           s     h     a     r     e     d
obj(240),c%(240),r%(240),zzz(120,5),tz(360),ttz(360),shiftpts%(5)
d   i     m           s     h     a     r     e     d
order%(240),slope(120,120),zz(120,120),xb%(2,120),bb%(2,5),jff%(5
),jee%(5),filenm$(5)
dim shared colis%(8)
dim shared m(480),mp(480)
d   i     m           s     h     a     r     e     d
z(150),h(9),ish(9),xh(5),yh(5),im(9),x(120),y(120),jm(5),castab(3
,3,3)
rem $static
d                       a                       t                       a
0,0,8,0,0,2,5,1,7,6,9,2,3,4,5,6,0,3,4,7,1,0,1,8,4,3,0,9,0,1,2,3,9
,6,7,4,5,2,0,5,8,0,0,6,7,8,9,1,3,5,0,3,8,0,0,3,8,0,3,6,9,0,1,1,0

screen 9

call printmenu
CALL initial
DO UNTIL selkey = 27                                   'ESC key to exit
program
   CALL kbhit                                     'wait for key hit from
operator
     SELECT CASE selkey
     CASE 59                                       'have the stage moved
to each
       CALL send("MR+18750")                       'aperture for testing
scanning
       call stopped
       CALL send("MR+18750")
       call stopped
       CALL send("MR+18750")
```

```
        call stopped
        CALL send("MR+18750")
        call stopped
        CALL send("MR-75000")
        call stopped
        call printmenu
      CASE 60
        CLS
        LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #1
     "
        shell("acq 5")                         'call acq.exe to get the
image
        call waiting
        CALL send("MR+18750")
        call stopped
        LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #2
     "
        shell("acq 10")
        call waiting
        CALL send("MR+18750")
        call stopped
        LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #3
     "
        shell("acq 15")
        call waiting
        CALL send("MR+18750")
        call stopped
        LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #4
     "
        shell("acq 20")
        call waiting
        CALL send("MR+18750")
        call stopped
        LOCATE 12,10
print "                   ACQUISITION SUBAPERTURE #5
     "
        shell("acq 25")
        call waiting
        CALL send("MR-75000")
        call stopped
        call main
        call printmenu
      CASE ELSE
      END SELECT
    LOOP
cls
END

'**
'*** Dummy waiting subroutine.
```

```
'**
sub waiting static
    for ii = -32767 to 32767
     for jj = 0 to 1
         next jj
     next ii
end sub


'**
'*** main process, contains computation , and display.
'**
sub main static
call init
cls
locate 12,1
print "                                         COMPUTING ...
                         "
call waiting
call display
end sub


'**
'*** wait for asterisk to make sure command is accept by controller
'**
SUB ASTERISK STATIC
    DO UNTIL INPUT$(1, #6) = "*"
       LOOP
END SUB

'**
'*** initial process
'**
SUB initial STATIC
OPEN "COM2:9600,N,8,1,RS,CS,DS,CD" FOR RANDOM AS #6  'OPEN SERIAL
PORT
CALL send("MED")                                 'disable mnemonics
CALL send("MFE")                                 'enable move finish
CALL send("VI600")                               'set initial velocity
CALL send("VF6000")                        'set final velocity
CALL send("AC1001")                                   'set acceleration and
deceleration
END SUB

'**
'*** accept key hit from keyboard
'**
SUB kbhit STATIC
   key$ = ""
   DO
   WHILE key$ = ""
      key$ = INKEY$
      WEND
   IF LEN(key$) > 1 THEN
      skey$ = MID$(key$, 2, 1)
     ELSE
      skey$ = MID$(key$, 1, 1)
      END IF
   selkey = ASC(skey$)
   LOOP UNTIL key$ <> ""
END SUB

'**
'*** subroutine print the operation menu onto screen
'**
SUB printmenu STATIC
CLS
LOCATE 3,1
print "                        CYLINDRICAL MIRROR MULTITRACE MEASUREMENT
"
print ""
print ""
print ""
print ""
print "                               F1 : TEST MIRROR SCANNING"
print ""
print "                               F2 : ACQUISITION AND MEASUREMENT"
print ""
print ""
print "                               ESC: EXIT"
END SUB

'**
'*** subroutine for sending command to controller
'**
SUB send (a$) STATIC
    PRINT #6, a$                                        'send command to
controller
    CALL ASTERISK                                   'wait until asterisk
is return
END SUB

'**
'*** wait until process finish
'**
SUB STOPPED STATIC
    DO UNTIL INPUT$(1, #6) = "F"
       LOOP
      for ii = -32767 to 32767
       for jj = 0 to 1
           next jj
       next ii
END SUB


'**
'*** initialize parameter.
```

```
'**
sub init static
    colis%(1) = 10
    colis%(2) =  3
    colis%(3) =  9
    colis%(4) =  1
    colis%(5) =  4
    colis%(6) = 12
    colis%(7) = 14
    colis%(8) = 15
    im(0)=0 : im(1)=1 : im(2)=1 : im(3)=0
    jm(0)=0 : jm(1)=0 : jm(2)=1 : jm(3)=1
    for k=0 to 3
     for j=0 to 3
         for i=0 to 3
          read castab(k,j,i)
          next i
         next j
     next k
open "demor.dat" for input as #1
open "demoi.dat" for input as #2
input #1,top%,bot%,maxp,minp
input #2,top%,bot%,maxi,mini
for i% = 1 to 120
    for j% = 1 to 120
     zz(j%,i%) = -32767
     slope(j%,i%) = -32767
     next j%
    next i%
sump = 0 : sumi = 0 : cntp = 0 : cnti = 0
for i% = top% to bot%
    input #1,dmy%,xb%(1,i%),xb%(2,i%)
    input #2,dmy%,xb%(1,i%),xb%(2,i%)
    for j% = xb%(1,i%) to xb%(2,i%)
     input #1,slope(j%,i%)
     input #2,zz(j%,i%)
     if slope(j%,i%) <> -32767 then
        sump = sump + slope(j%,i%)^2
        cntp = cntp + 1
        end if
     if zz(j%,i%) <> -32767 then
        sumi = sumi + zz(j%,i%)^2
        cnti = cnti + 1
        end if
     next j%
    next i%
rmsp = sqr(sump / cntp)
rmsi = sqr(sumi / cnti)
close
end sub

'**
'*** initialize parameter of slope display.
'**
```

```
sub initp static
    title$="CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE
FUNCTION"
    levell% = 12
    zscale = -.3
    deginc = 36
    mark% = 60
    for i% = 1 to 480
     m(i%) = 32767
     mp(i%) = 32767
      next i%
    max = maxp
    min = minp
    rms = rmsp
    pv = maxp - minp
end sub

'**
'*** initialize parameter of wavefront display.
'**
sub initi static
    title$="CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT
FUNCTION"
    levell% = 180
    zscale = -.005
    deginc = 720
    mark% = 1800
    for i% = 1 to 480
     m(i%) = 32767
     mp(i%) = 32767
     next i%
    max = maxi
    min = mini
    rms = rmsi
    pv = maxi - mini
end sub



'**
'*** display the result.
'**
sub display static
    call initp
    call plot(slope())
    ys = 2 : xs = 4.5 : xsp =  (640 - xs * 120)/2 : ysp = 50 - top
* ys
    call contour(slope(),lll%,rrr%,top%,bot%,xs,ys,xsp,ysp)
    key$ = ""
    WHILE key$ = ""
       key$ = INKEY$
       WEND
    call initi
    call plot(zz())
    ys = 2 : xs = 4.5 : xsp =  (640 - xs * 120)/2 : ysp = 50 - top
* ys
    call contour(zz(),lll%,rrr%,top%,bot%,xs,ys,xsp,ysp)
    key$ = ""
    WHILE key$ = ""
       key$ = INKEY$
       WEND
end sub


'**
'*** hidden line remove routine.
'**
sub hiddenline(x1,x2,y1,y2,m1,m2,mm1,mm2,col%) static

        '-------  check the hidden line ----------------
      if y1<m1 then
         if y2<m2 then
            mm1=y1
            mm2=y2
            line (x1,y1)-(x2,y2),colis%(col%)
         else
            m=(y2-y1)/(x2-x1)
            b=y1-m*x1
            d=(y1-m1)*(x2-x1)/(m2-y2+y1-m1)
            x=x1+d
            y=m*x+b
            line (x1,y1)-(x,y),colis%(col%)
            mm1=y1
            mm2=m2
         end if
      elseif y2<m2 then
         m=(y1-y2)/(x1-x2)
         b=y1-m*x1
         d=(m1-y1)*(x2-x1)/(y2-m2+m1-y1)
         x=x1+d
         y=m*x+b
         line (x,y)-(x2,y2),colis%(col%)
         mm1=m1
         mm2=y2
      else
         mm1=m1
         mm2=m2
      end if
end sub

'**
'*** 3 Dimensional hidden line removed display
'**
sub plot(arr(2)) static
    cls
    county% = 0
    for i% = bot% + stepr% to top% step -stepr%
     prehigh = 0
     for j%=111% to rrr% step 1
         if arr(j%,i%) <> -32767 then
            high = arr(j%,i%)-min
            y2=high*zscale-county%*yshift+ydc
           else
            y2=-county%*yshift+ydc
            high = 0
            end if
         if high <> 0 then
            coli% = high / levell%
            coli% = coli% mod 7 + 1
           else
            if prehigh <> 0 then
            coli% = prehigh / levell%
            coli% = coli% mod 7 + 1
           else
            coli% = 8
           end if
           end if
         x2=j%*xscale+xdc
         if county%=0 then
           mp(j0%)=y2
           if j% > 111% then line (x1,y1)-(x2,y2),colis%(coli%)
          else
           if j% > 111% then
           max1=m(j%-1)
           max2=m(j%)
           call hiddenline(x1,x2,y1,y2,max1,max2,maxp1,maxp2,coli%)
           mp(j%-1)=maxp1
           mp(j%)=maxp2
           end if
           end if
        x1=x2:y1=y2
        prehigh = high
        next j%
     county% = county% + 1
     next i%
    for j%=1 to 480
     m(j%)=mp(j%)
    next j%
    call drawmark
    call boxf
    call sec(arr())
end sub

'**
'*** contour display
'**
sub contour (arrd(2),xf%,xe%,yf%,ye%,xs,ys,xsp,ysp) static
cls
for i = 1 to 120
    x(i) = xsp + (i - 1) * xs
    next i
for i = 1 to 120
```

```
    y(i) = ysp + (i - 1) * ys
    next i

if min <= 0 and max >= 0 then
   exlevel = 1
  else
   exlevel = 0
   end if

locate 22,64
print level1% chr$(248);"/ LEVEL"
call box1

mainp:
for i = 0 to 150
    z(i) = min + i * level1%
    if z(i) > max then
       level = i
       i = 200
       end if
    next i

for j=ye% - stpno% to yf% step -stpno%
    if j<top% or j>=bot% then goto noneinbox1
    for i=xf% to xe% - stpno% step stpno%
     if i<xb%(1,j) or i>xb%(2,j) - 1 or i < xb%(1,j+stpno%) or i >
xb%(2,j+stpno%) - 1 then goto noneinbox
     if  arrd(i,j)  =  -32767  or  arrd(i+stpno%,j)  =  -32767  or
arrd(i,j+stpno%) = -32767 or arrd(i+stpno%,j+stpno%) = -32767 then
goto noneinbox
     locate 23,10
     if (arrd(i,j)<arrd(i,j+stpno%)) then
        dmin=arrd(i,j)
     else
        dmin=arrd(i,j+stpno%)
     end if
     if arrd(i+stpno%,j)<dmin then dmin=arrd(i+stpno%,j)
     if   arrd(i+stpno%,j+stpno%)<dmin   then
dmin=arrd(i+stpno%,j+stpno%)
     if arrd(i,j)>arrd(i,j+stpno%) then
        dmax=arrd(i,j)
     else
        dmax=arrd(i,j+stpno%)
     end if
     if arrd(i+stpno%,j)>dmax then dmax=arrd(i+stpno%,j)
     if   arrd(i+stpno%,j+stpno%)>dmax   then
dmax=arrd(i+stpno%,j+stpno%)
     if dmax<z(0) or dmin>z(level-1) then goto noneinbox
     for k=0 to level - 1
         if z(k)<dmin or z(k)>dmax then goto noneintri
         for m=4 to 0 step -1
          if m>0 then
             h(m)=arrd(i+stpno%*im(m-1),j+stpno%*jm(m-1))-z(k)
             xh(m)=x(i+stpno%*im(m-1))
         yh(m)=y(j+stpno%*jm(m-1))
      end if
      if m=0 then
         h(0)=(h(1)+h(2)+h(3)+h(4))/4
         xh(0)=(x(i)+x(i+stpno%))/2
         yh(0)=(y(j)+y(j+stpno%))/2
      end if
      if h(m)>0 then
         ish(m)=2
      elseif (h(m)<0) then
         ish(m)=0
      else
         ish(m)=1
      end if
     next m
     for m=1 to 4
      m1=m:m2=0:m3=m+1
      if m3=5 then m3=1
      casetype=cint(castab(ish(m1),ish(m2),ish(m3)))
        if casetype=0 then
        goto case0
        end if
      o   n   c   a   s   e   t   y   p   e   g   o   t   o
case1,case2,case3,case4,case5,case6,case7,case8,case9
      case1:
           x1=xh(m1):y1=yh(m1):x2=xh(m2):y2=yh(m2)
           goto drawit
      case2:
           x1=xh(m2):y1=yh(m2):x2=xh(m3):y2=yh(m3)
           goto drawit
      case3:
           x1=xh(m3):y1=yh(m3):x2=xh(m1):y2=yh(m1)
           goto drawit
      case4:
           x1=xh(m1):y1=yh(m1)
           x2=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
           y2=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
           goto drawit
      case5:
           x1=xh(m2):y1=yh(m2)
           x2=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
           y2=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
           goto drawit
      case6:
           x1=xh(m3):y1=yh(m3)
           x2=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
           y2=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
           goto drawit
      case7:
           x1=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
           y1=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
           x2=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
           y2=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
           goto drawit
```

```
          case8:
               x1=(h(m3)*xh(m2)-h(m2)*xh(m3))/(h(m3)-h(m2))
               y1=(h(m3)*yh(m2)-h(m2)*yh(m3))/(h(m3)-h(m2))
               x2=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
               y2=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
               goto drawit
          case9:
               x1=(h(m1)*xh(m3)-h(m3)*xh(m1))/(h(m1)-h(m3))
               y1=(h(m1)*yh(m3)-h(m3)*yh(m1))/(h(m1)-h(m3))
               x2=(h(m2)*xh(m1)-h(m1)*xh(m2))/(h(m2)-h(m1))
               y2=(h(m2)*yh(m1)-h(m1)*yh(m2))/(h(m2)-h(m1))
           drawit:
             x1 = int(x1)
             x2 = int(x2)
             y1 = int(y1)
             y2 = int(y2)
             line (x1,y1)-(x2,y2)
          case0: next m

noneintri: next k
noneinbox: next i
noneinbox1:  next j
10202 end sub


'**
'*** draw scale index
'**
sub drawmark static
    unitx = 535 : unity = 45
    unitsize = mark% * zscale

  line(unitx,unity)-(unitx,unity-unitsize)
  line(unitx-3,unity)-(unitx+3,unity)
  line(unitx-3,unity-unitsize)-(unitx+3,unity-unitsize)
  locate 4,70
  print mark%;chr$(248) ;
end sub



'**
'*** outline box
'**
sub boxf static
  locate 1,8
  print title$
  line(0,0)-(639,349),,b
  line(3,17)-(636,346),,b
  line(0,14)-(639,14)
  locate 14,50
  print "RMS";rms ;
  locate 14,65
  print "P-V";pv ;
```

```
end sub
'**
'*** outline box
'**
sub box1 static
  locate 1,8
  print title$
  line(0,0)-(639,349),,b
  line(3,17)-(636,346),,b
  line(0,14)-(639,14)
  locate 23,65
  print "RMS";rms ;
  locate 24,65
  print "P-V";pv  ;
end sub

'**
'*** sectional display routine
'**
sub sec(arr(2)) static
  yff = 290
  yee = 200
  mmff = 0
  xff = 111% * xscale + xdc
  xee = rrr% * xscale + xdc
  mmffxc = (xee - xff) / (rrr% - 111%) * 1.190625
  degsc = 85 / (max - min + deginc * .99)

  if top% mod 5 then
     ydfr = (top% / 5 + 1) * 5
    else
     ydfr = top%
     end if
  ydlr = bot% / 5 * 5

  for i% = ydfr to ydlr step 20
      xx = (rrr% + 5) * xscale + xdc
      yy = (i% - bot%) / stepr% * yshift + ydc
      line(xx,yy)-(xx+15,yy)
      row% = yy /14 + 1
      col% = xx / 8 + 3
      locate row%,col%
      print i% ;
      next i%
  secno% = 0
  do while secno% <> 999
     secno% = 0
     do until secno% = 999 or secno% >= top% and secno% <= bot%
     locate 23,5
     print "Key in trace no.";top%;"-";bot%; " 999 to quit " ;"
                                         " ;
     locate 23,50
     input secno%
     if secno% <> 999 and (secno% < top% or secno% > bot%) then
```

```
print chr$(7) ;
     loop
     if secno% <> 999 then
     call clearsec
     locate 23,35
     print "TRACE #";secno%
     locate 22,6
     print "(mm)" ;
     for degff=0 to (max-min) + deginc *.99 step deginc
         yy = -degff * degsc + yff
         line(xff,yy)-(xee,yy),,,&hcccc
         locate yy / 14 + 1,68
         print degff;chr$(248) ;
         next degff
     for mmff = 0 to 99 step 10
         xx = mmff * mmffxc + xff
         line(xx,yff)-(xx,yy),,,&hcccc
         mmm% = mmff
         if mmm% mod 20 = 0 then
            loc-te yff/14+1,xx/8
            print mmm% ;
            end if
         next mmff
     line(xee,yff)-(xee,yy) ,,,&hcccc

     first = 1
     sum = cnt = 0
     peak = -32767   : valley = 32767
     for j% = lll% to rrr%
         if arr(j%,secno%) <> -32767 then
            xx = j% * xscale + xdc
            yy = (min-arr(j%,secno%)) * degsc + yff
            if peak < arr(j%,secno%) then peak = arr(j%,secno%)
            if valley > arr(j%,secno%) then valley = arr(j%,secno%)
            sum = sum + arr(j%,secno%)^2
            cnt = cnt + 1
            if first <> 1 then line(xx,yy)-(xxo,yyo),14
            first = 0
            xxo = xx
            yyo = yy
            end if
         next j%
     locate 23,50
     print "RMS";sqr(sum/cnt) ;
     locate 23,65
     print "P-V";peak-valley ;
     key$ = ""
     WHILE key$ = ""
           key$ = INKEY$
           WEND
     end if
     loop
end sub
```

```
'**
'*** clean clipregin
'**
sub clearsec static
locate 15,2
print "
                 " ;
locate 16,2
print "
                 " ;
locate 17,2
print "
                 " ;
locate 18,2
print "
                 " ;
locate 19,2
print "
                 " ;
locate 20,2
print "
                 " ;
locate 21,2
print "
                 " ;
locate 22,2
print "
                 " ;
locate 23,2
print "
                 " ;
locate 24,2
print "
                 " ;
end sub
```

6.0 <u>TESTING AND EXPERIMENTAL RESULTS</u>

6.1 <u>The Test Mirror</u>

A cylindrical convex surface was custom fabricated for testing purposes. The surface is characterized as follows:

- Dimension: 3" wide x 9" long
- Radius of curvature: 300mm
- Surface errors: figure and macroroughness.

The mirror is mounted on a tilt/rotation manual stage for alignment purposes. This in turn is mounted on the NEAT scanning stage for subaperture testing.

6.2 <u>Subaperture Interferograms</u>

In the following, we present a number of subaperture interferograms for various parts of the mirror and various interferometer alignment conditions.

Three subapertures were chosen along the mirror surface:

- Subaperture 1: Medium surface quality
- Subaperture 2: good surface quality
- Subaperture 3: Poor surface quality.

For the subapertures, interferograms are captured with various interferometer alignment conditions as follows:

- Grating in focus and mirror aligned
- Grating in focus and mirror angularly misaligned by various amounts
- Grating out-of focus and mirror aligned
- Grating out-of focus and mirror angularly misaligned by various amounts.

Photographs of the various interferograms are shown in Figure 8 with self-explanatory captions.

6.3 Subaperture Testing: detailed Operation

In the following we present the results of the testing program showing the details of the subaperture testing approach.

The test makes use of 5 subapertures. Each subaprture is 2" long and the overlap area between subapertures is 1", so that the total scan length is about 6" long. For each subaperture the phase function, i.e. the wavefront slope function, is first independently computed. Then, simple first order least square fitting is carried out in the overlap area to create a continuous function along the mirror.

The results are given in Figures 9 to 12:

- Figure 9 shows photographs of the 5 subaperture interferograms, separate and side by side
- Figure 10 shows for each subaperture the phase measurement (wavefront slope function), displayed in the form of 3-D isometric plots and contour maps
- Figure 11 shows the effect of least square fitting in the overlap areas. The horizontal line indicates the boundary between two successive subapertures.
- Figure 12 shows the complete synthesized surface (slope function) displayed in the form of 3-D isometric plots and contour maps.

6.4 Subaperture Testing: Final Results

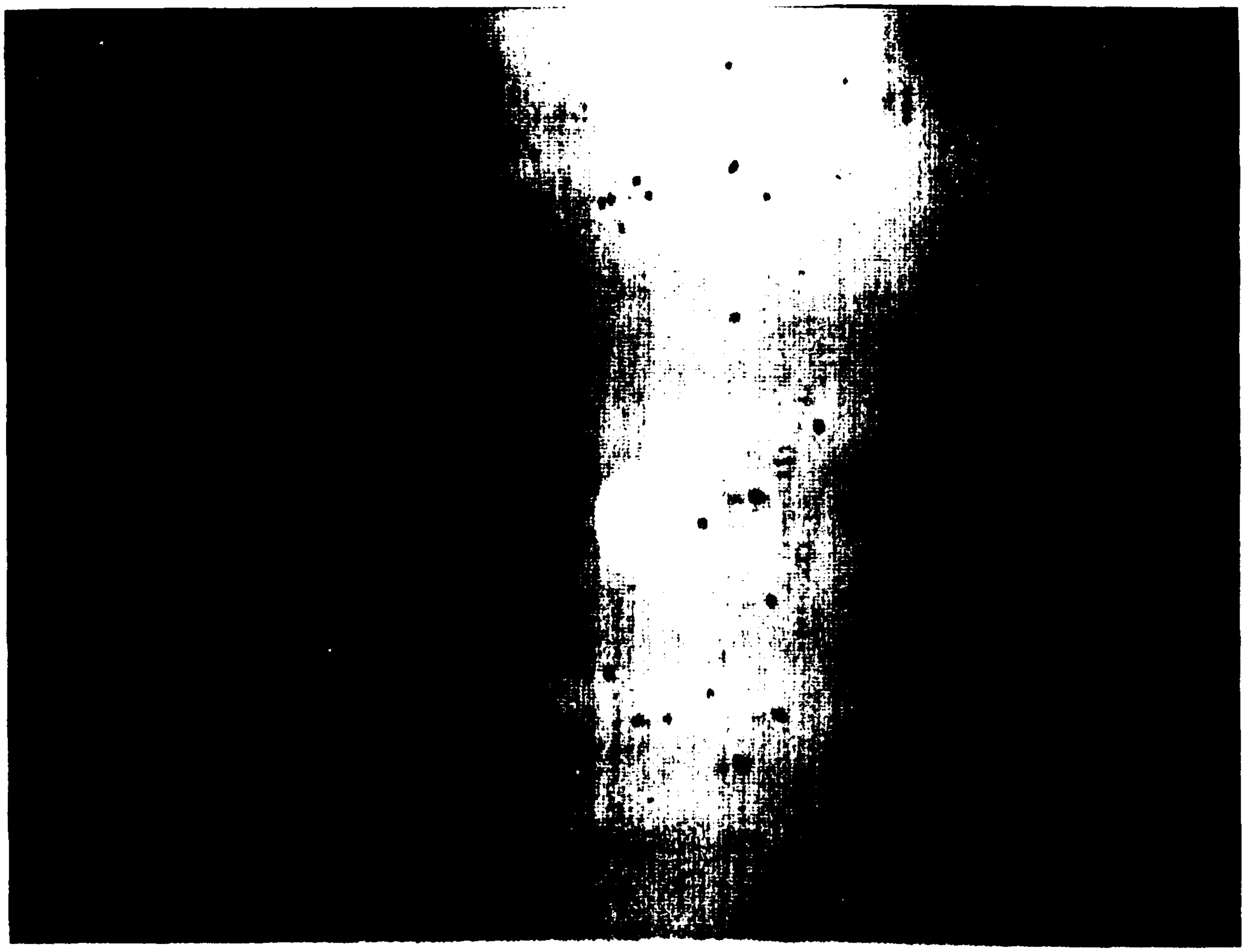
The FSIS system is characterized by fully automated operation, whereby successive subaperture interferograms are captured, computed and fitted to yield a full-surface, multi-trace wavefront measurement.

The raw phase measurement yields the wavefront slope and a numerical integration in the direction of the long side of the mirror yield a set of actual wavefront profiles or traces. Interactive processing permits to extract individual traces as desired.

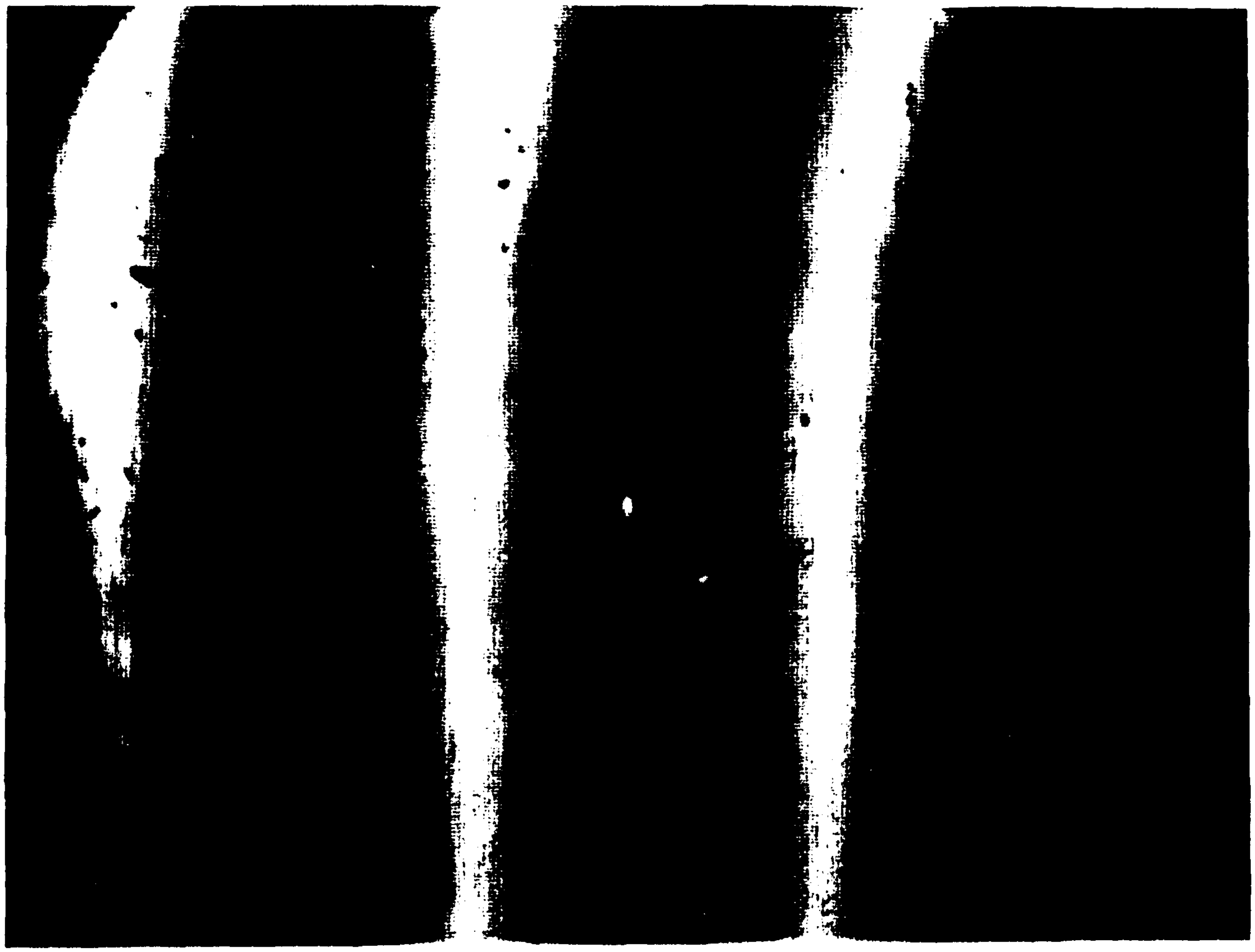
Figures 13 and 14 show the result of a complete measurement, yielding about 100 traces in parallel.

Note that, in Figures 11 to 14, the continuity of the contour lines indicates the continuity of the surface and thus the accuracy of the fit.

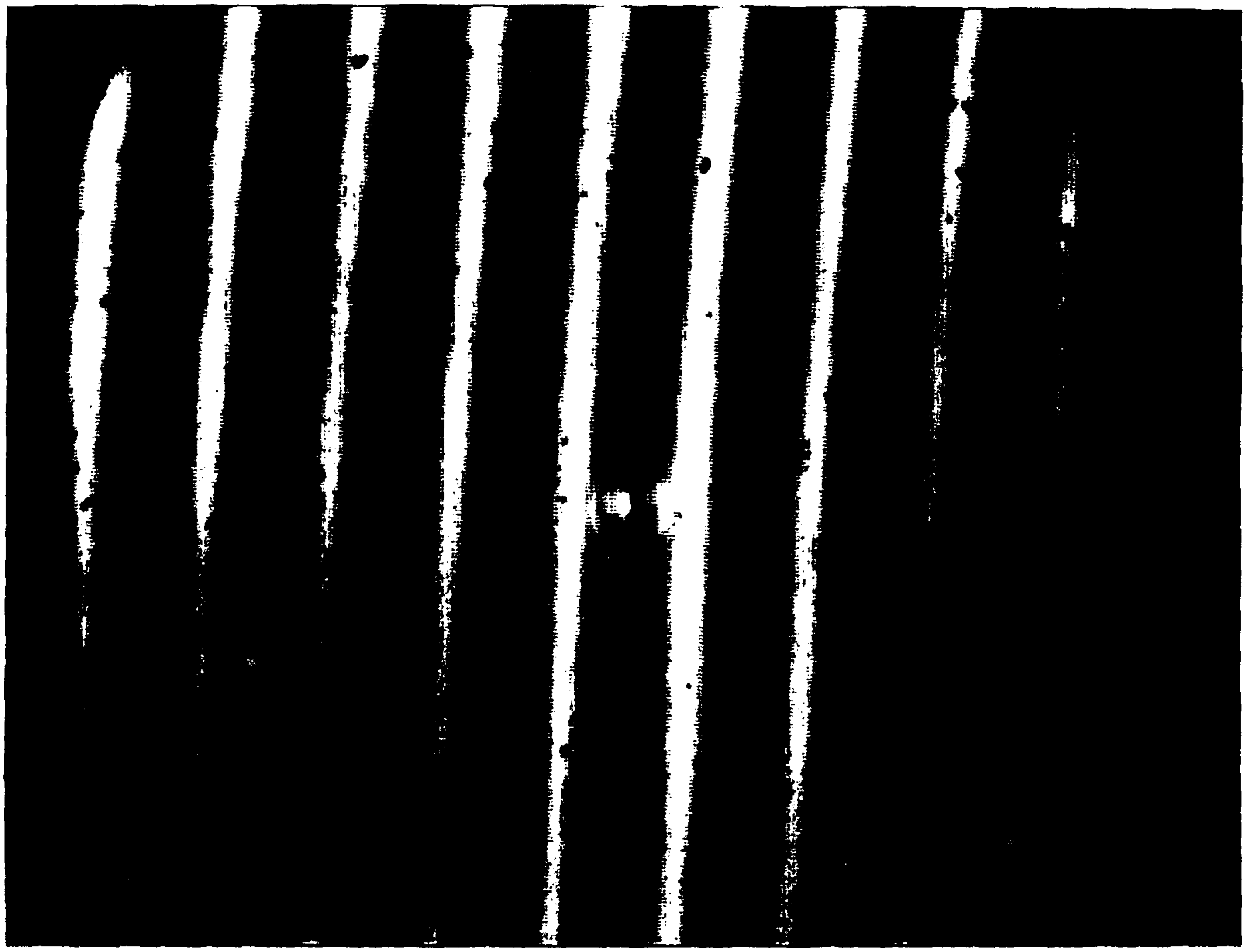
FIGURE 8: PHOTOGRAPHS OF INTERFEROGRAMS FOR 3 SUBAPERTURES FOR VARIOUS ALIGNMENT CONDITIONS



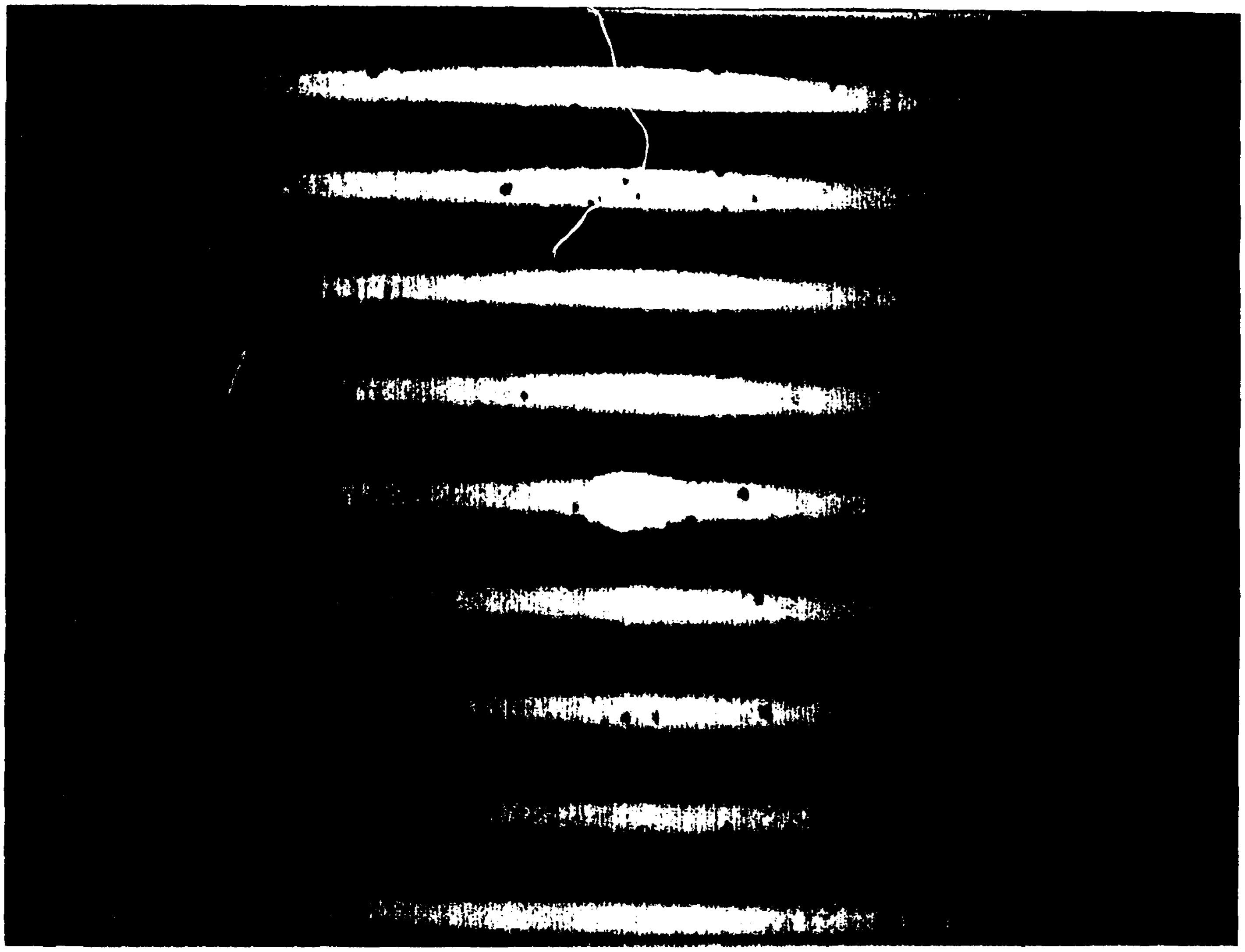
SUBAPERTURE 1: GRATING IN-FOCUS, MIRROR ALIGNED



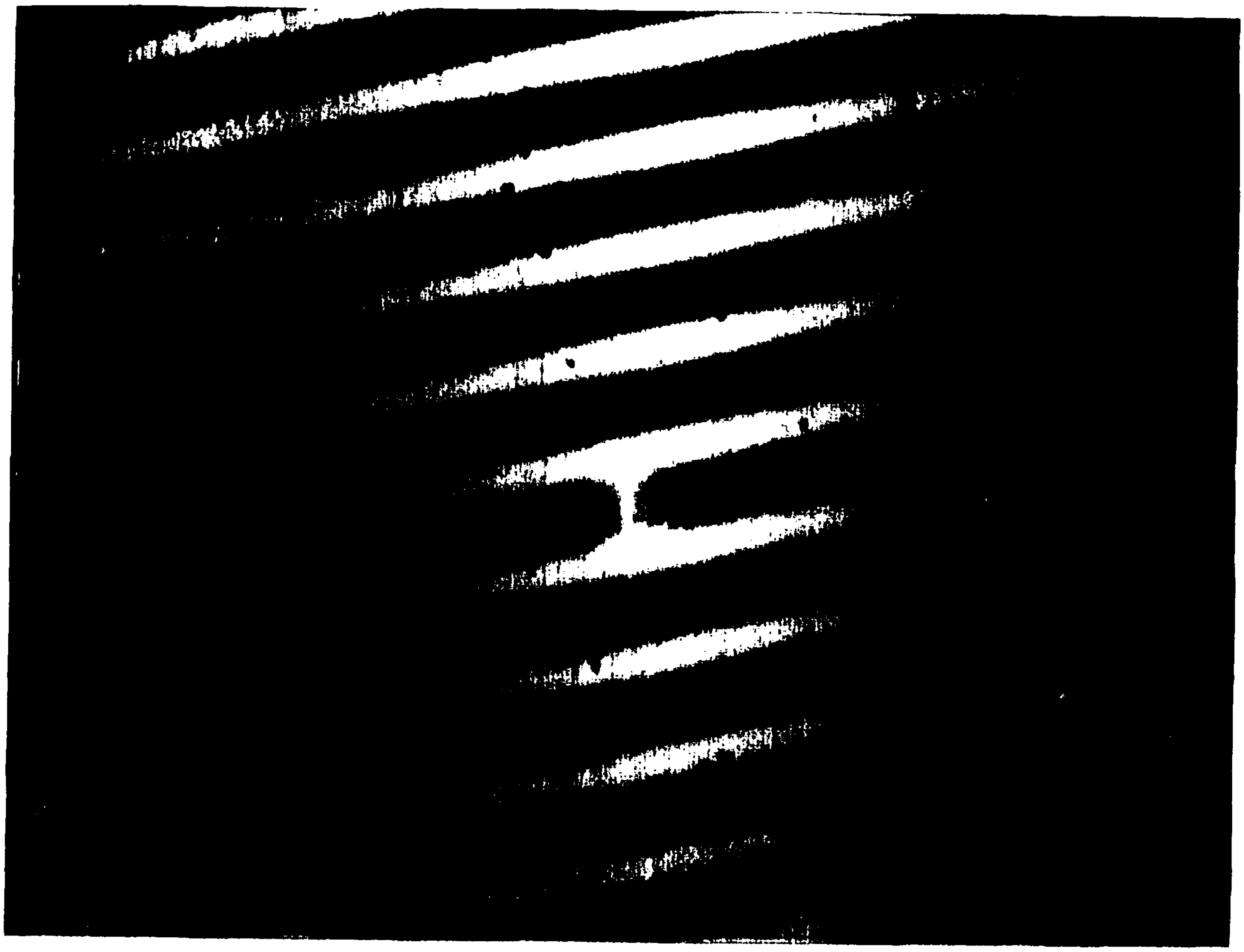
SUBAPERTURE 1: GRATING IN-FOCUS, MIRROR MISALIGNED



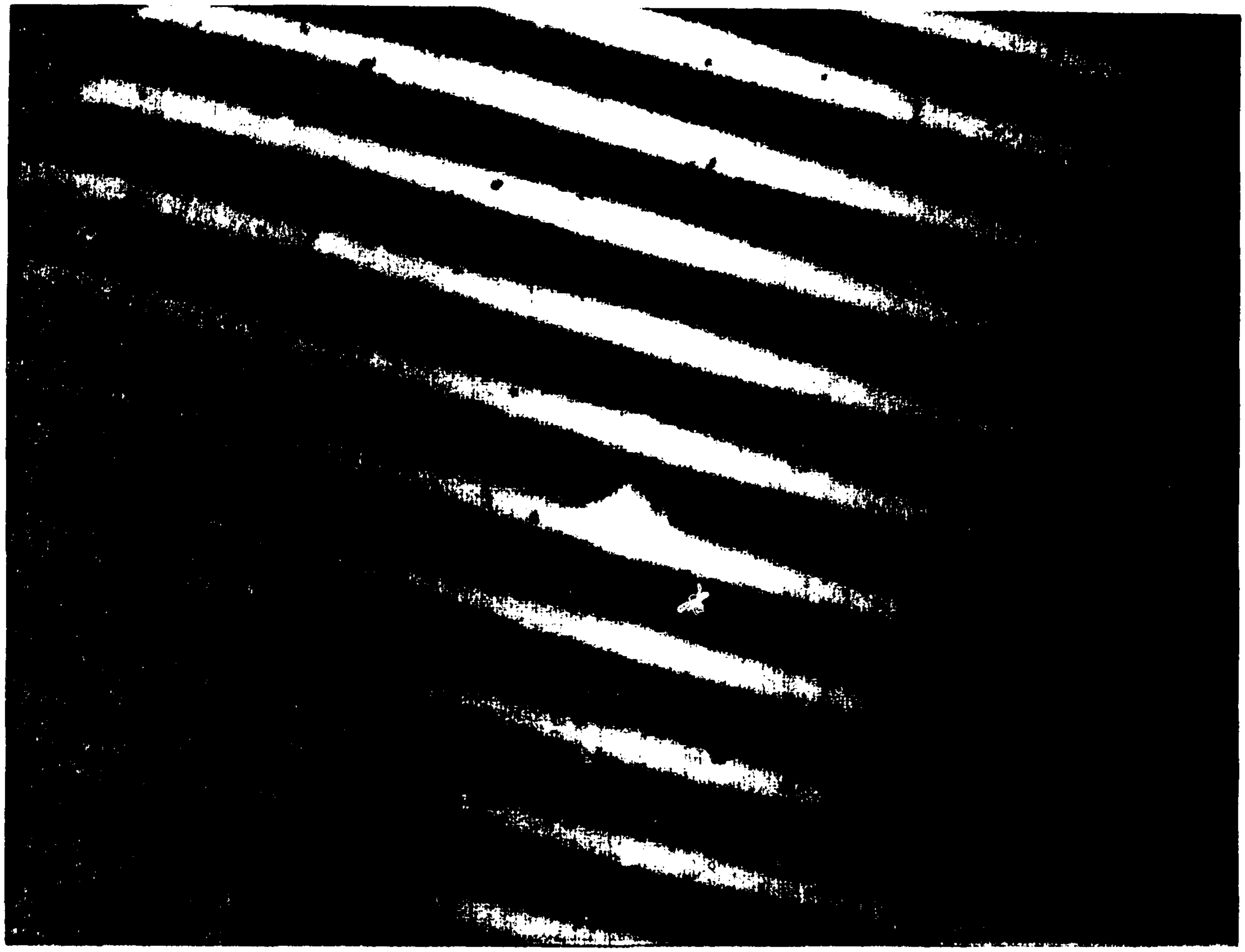
SUBAPERTURE 1: GRATING IN-FOCUS, MIRROR STRONGLY MISALIGNED



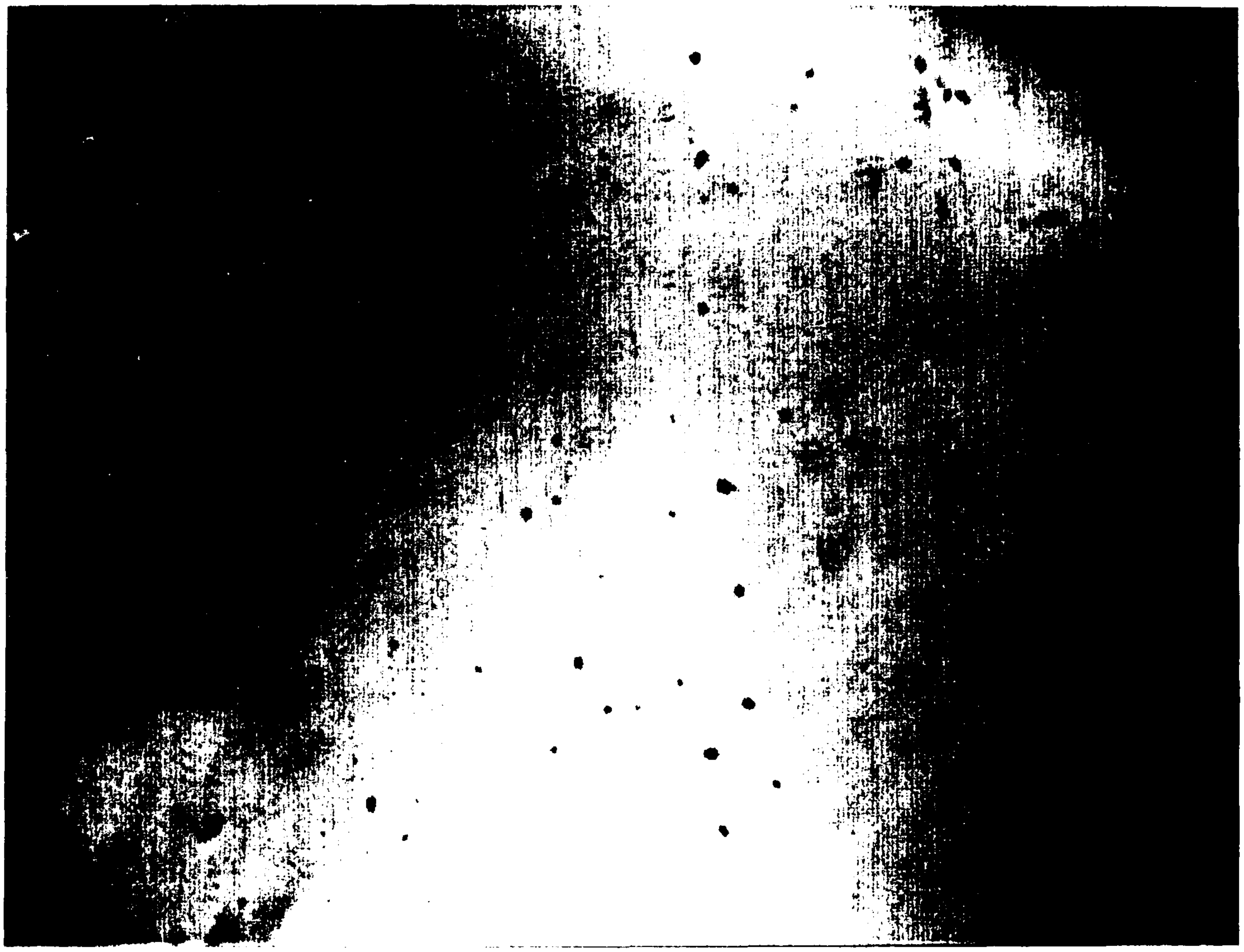
SUBAPERTURE 1: GRATING OUT-OF-FOCUS, MIRROR ALIGNED



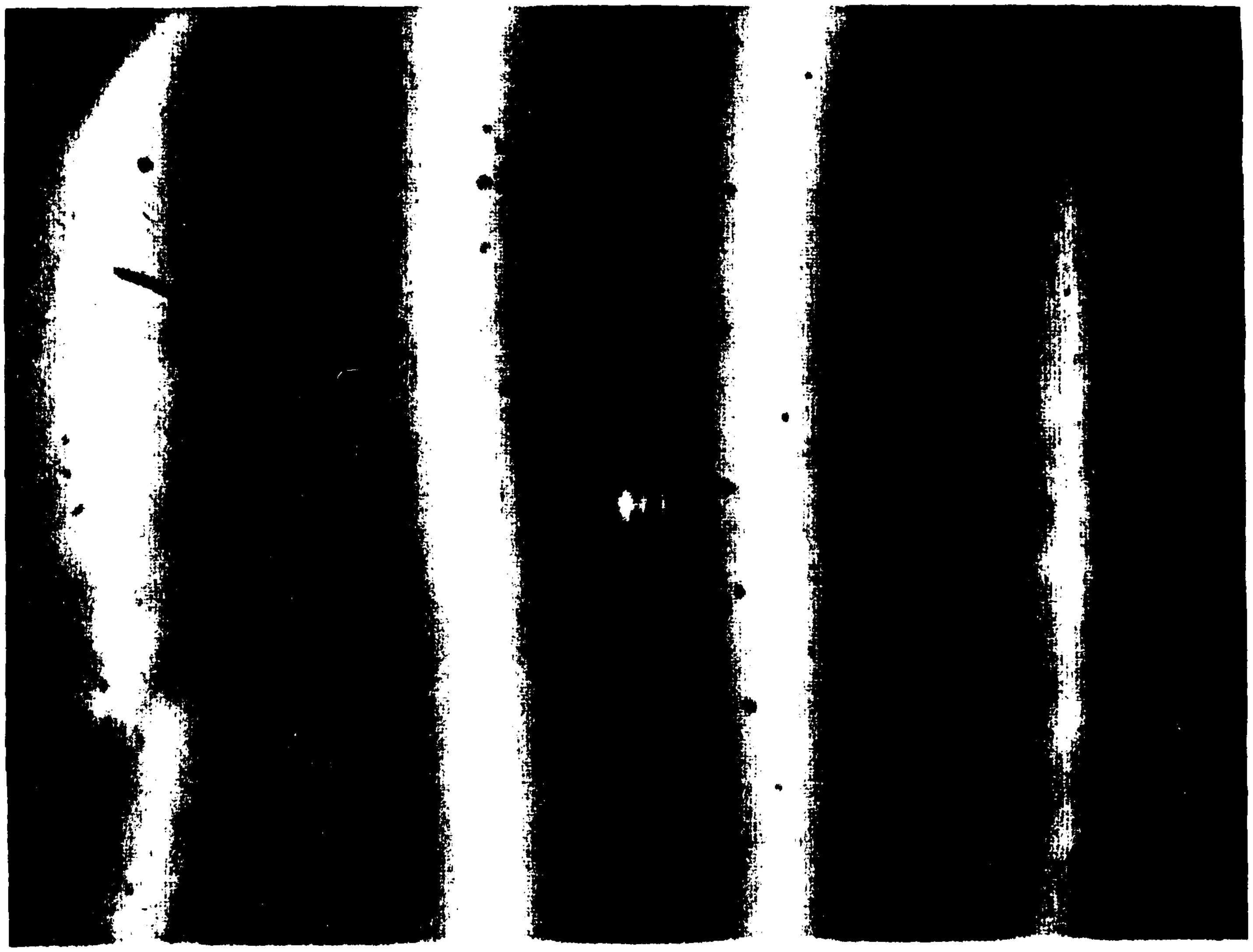
SUBAPERTURE 1: GRATING OUT-OF-FOCUS, MIRROR MISALIGNED (POSITIVE)



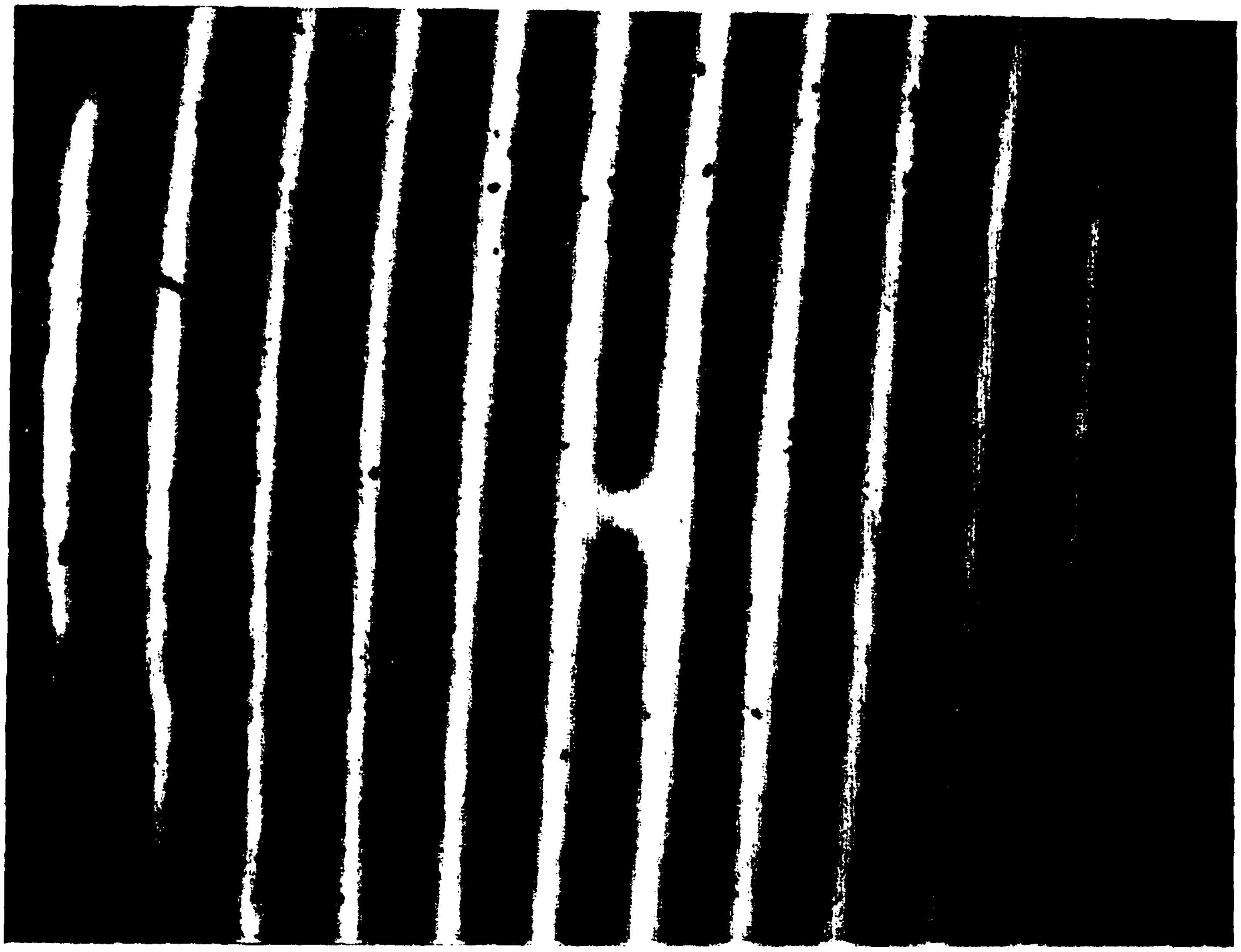
SUBAPERTURE 1: GRATING OUT-OF-FOCUS, MIRROR MISALIGNED (NEGATIVE)



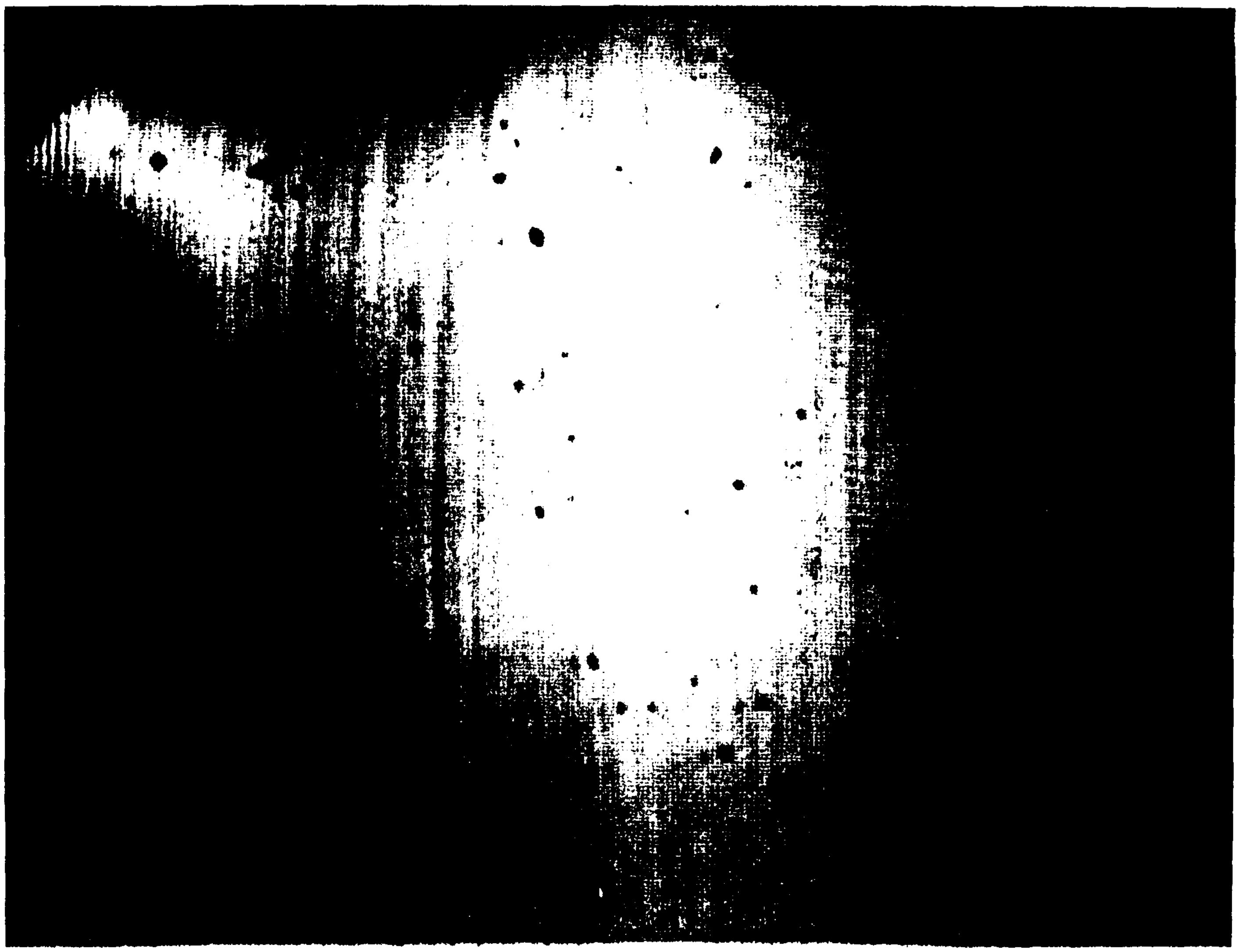
SUBAPERTURE 2: GRATING IN-FOCUS, MIRROR ALIGNED



SUBAPERTURE 2: GRATING IN-FOCUS, MIRROR MISALIGNED



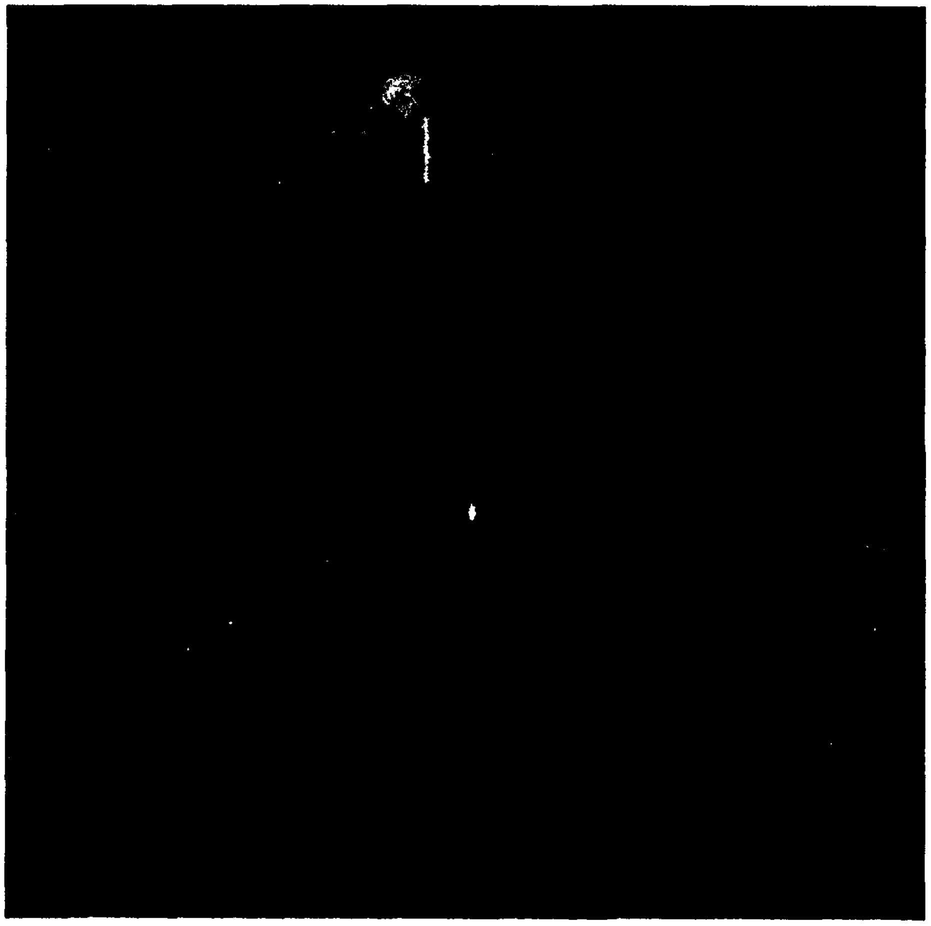
SUBAPERTURE 2: GRATING IN-FOCUS, MIRROR STRONGLY MISALIGNED



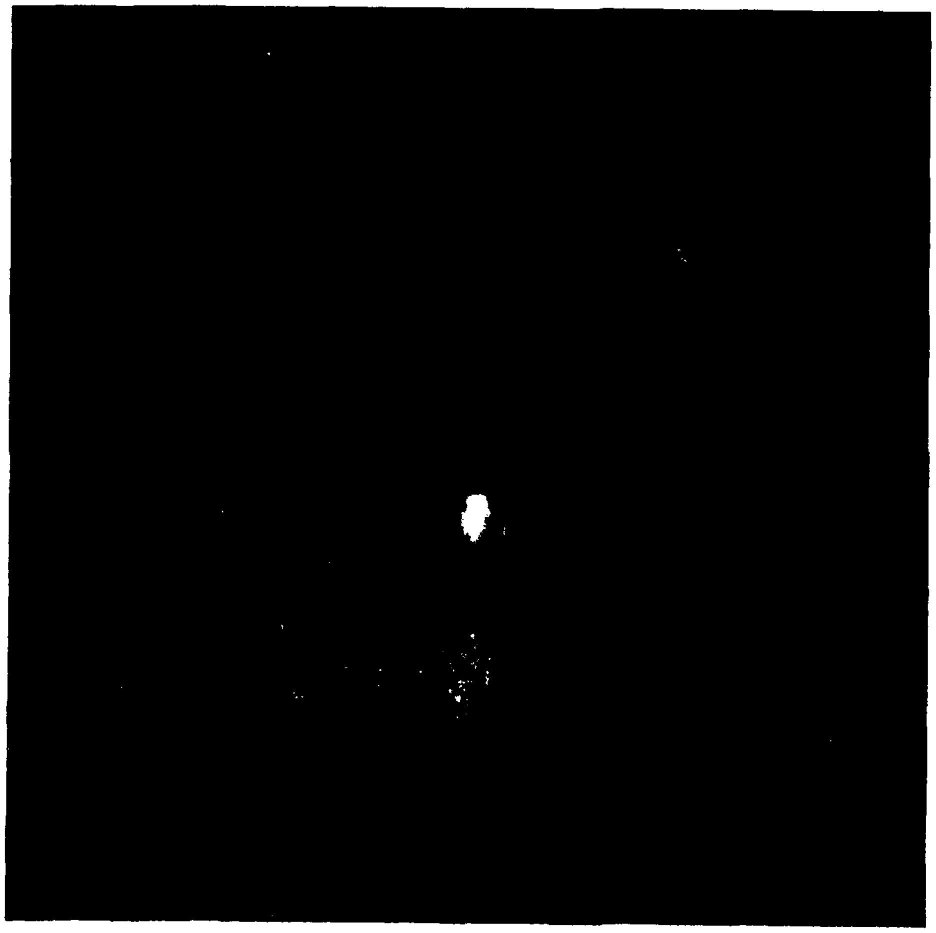
SUBAPERTURE 3: GRATING IN-FOCUS, MIRROR ALIGNED

FIGURE 9: PHOTOGRAPHS OF SUBAPERTURE INTERFEROGRAMS (5)

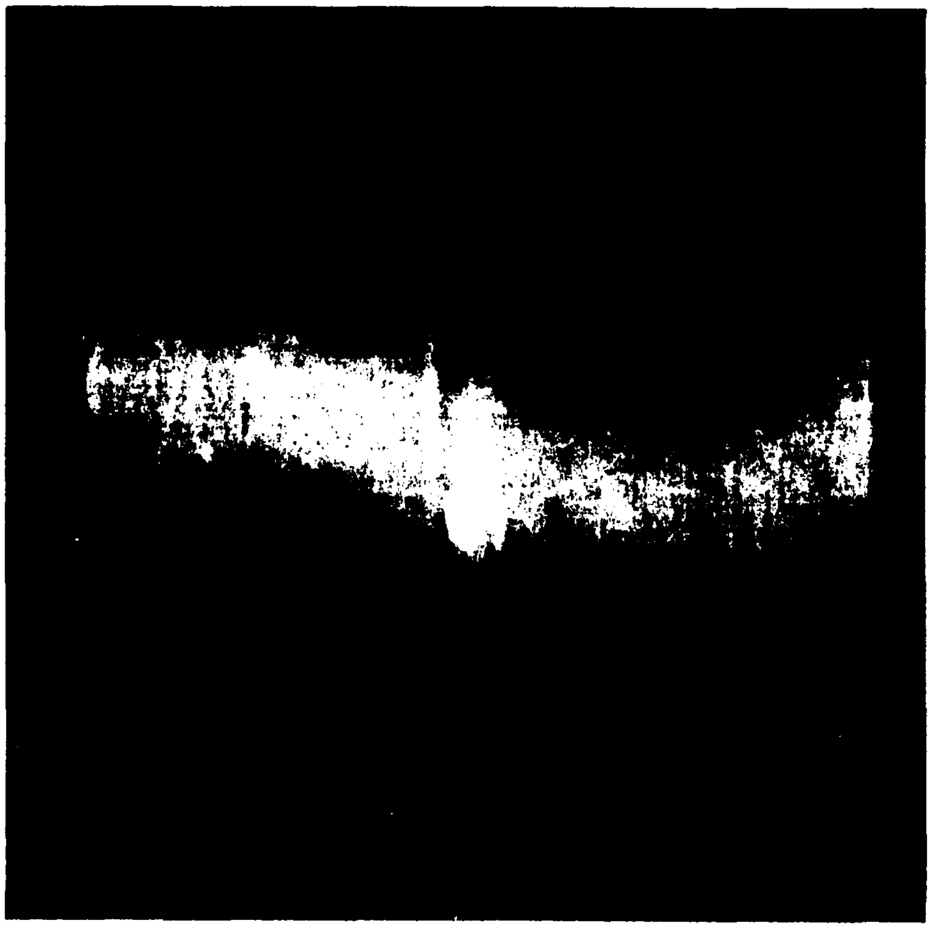
SUBAPERTURE 1



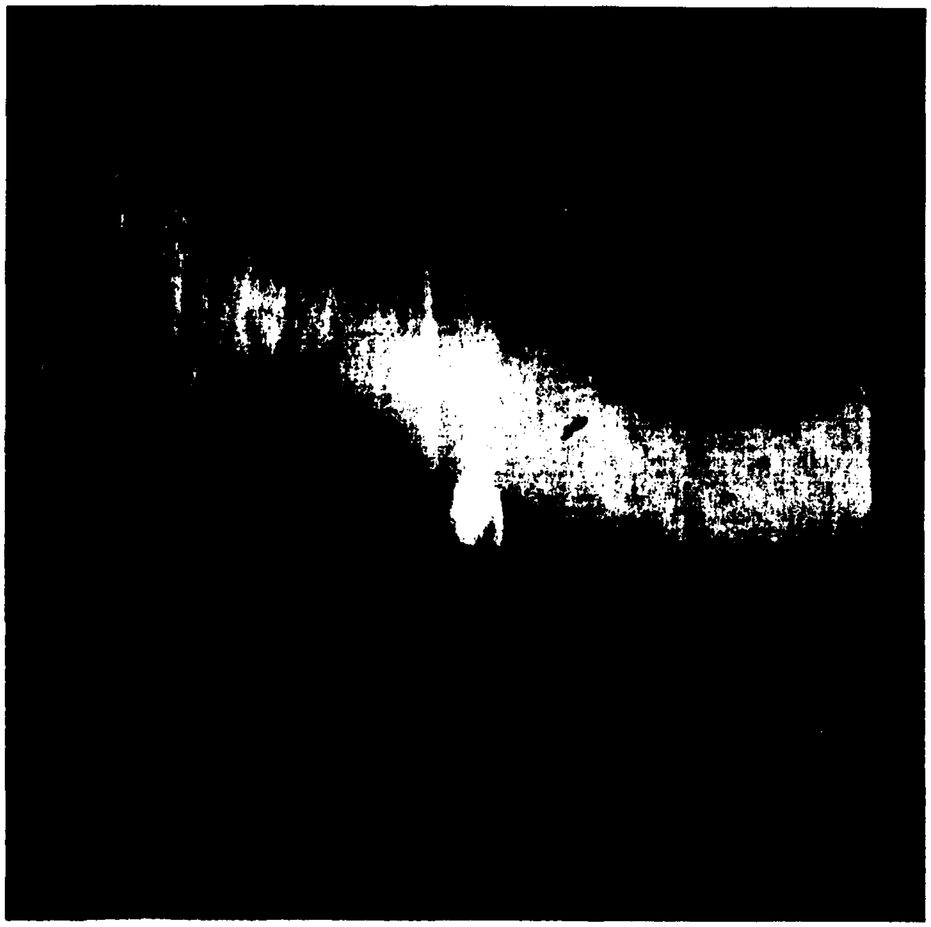
SUBAPERTURE 2



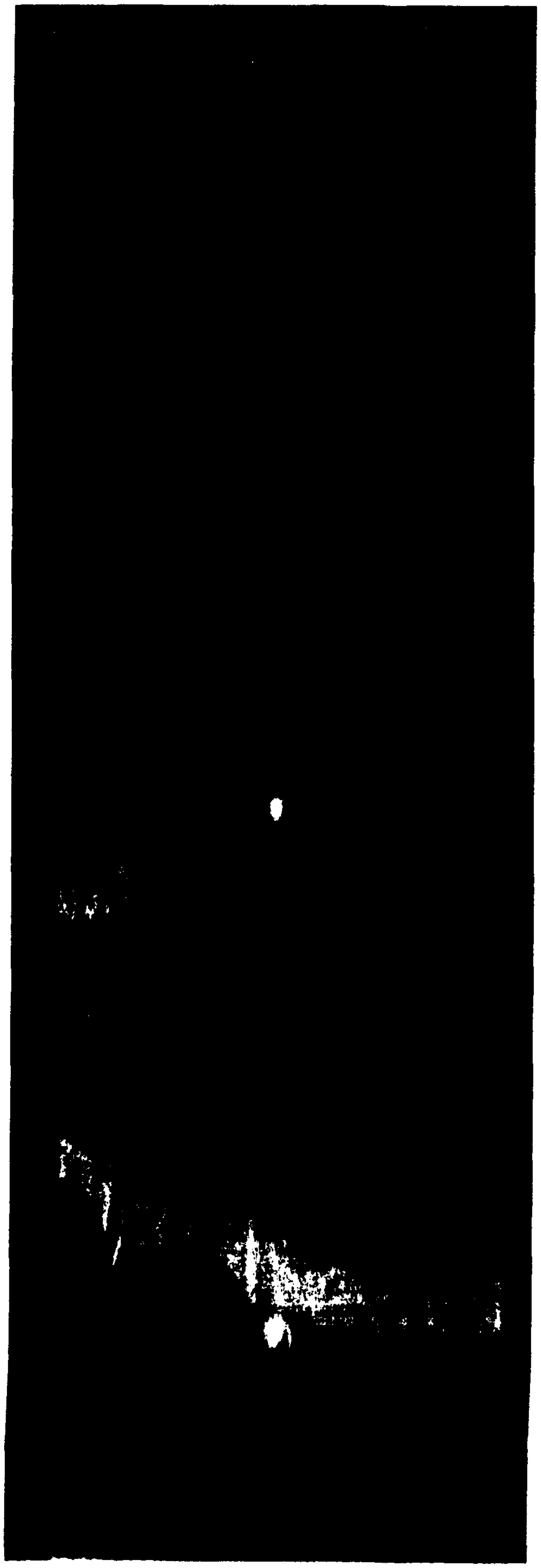
SUBAPERTURE 3

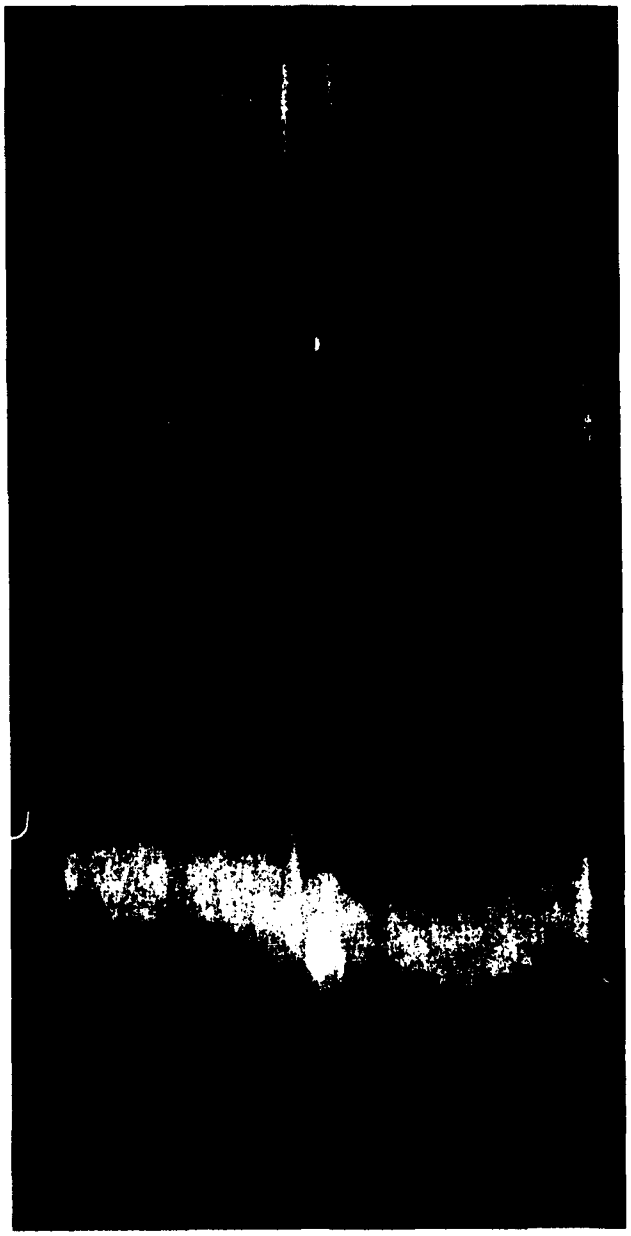


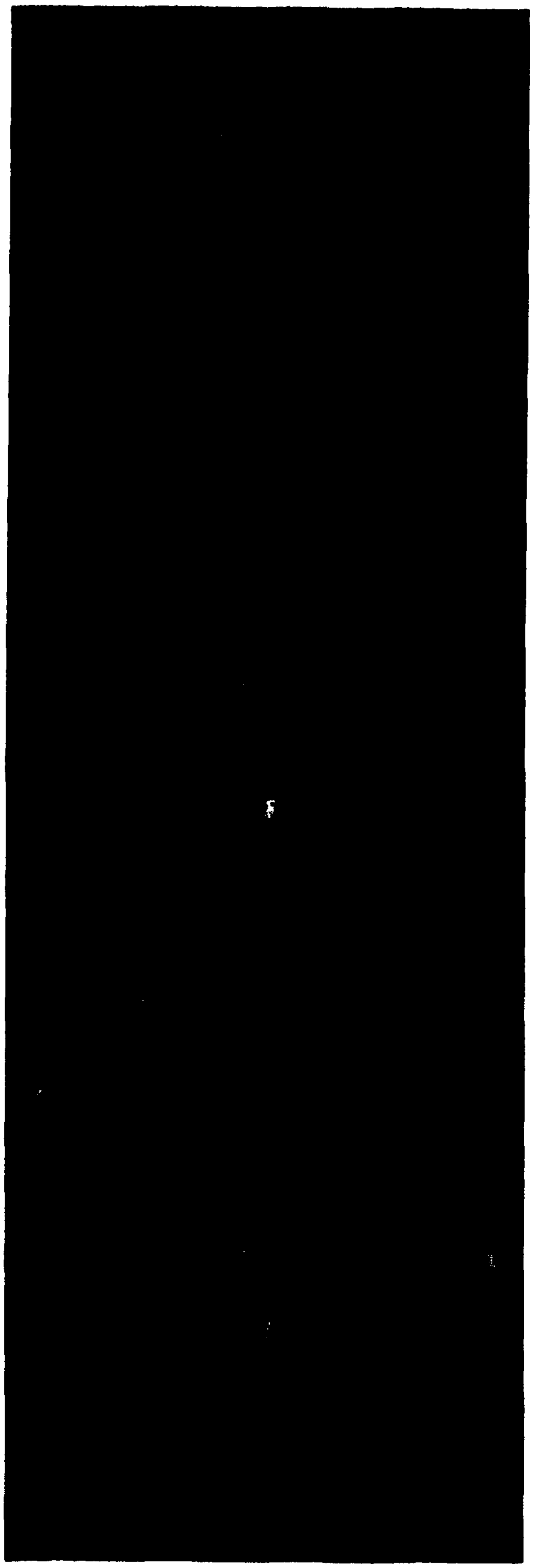
SUBAPERTURE 4



SUBAPERTURE 5


1
3
5


2
4

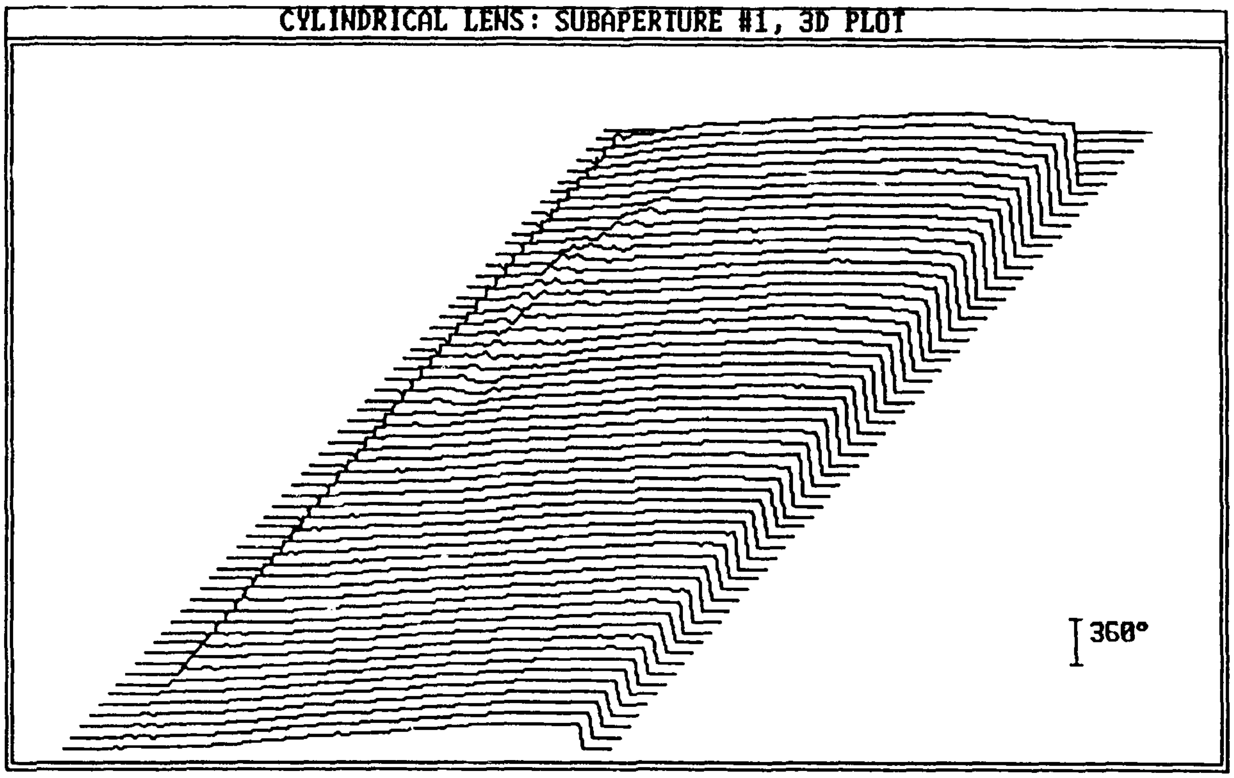


1

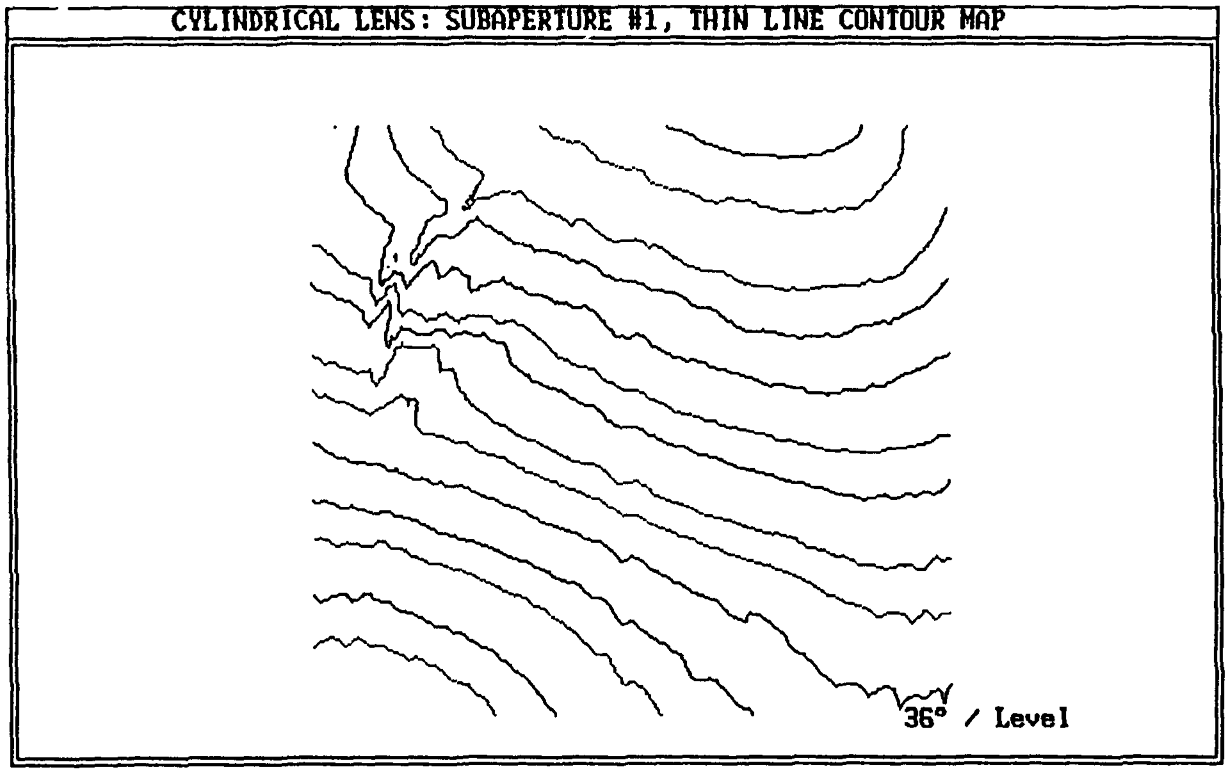
3

5

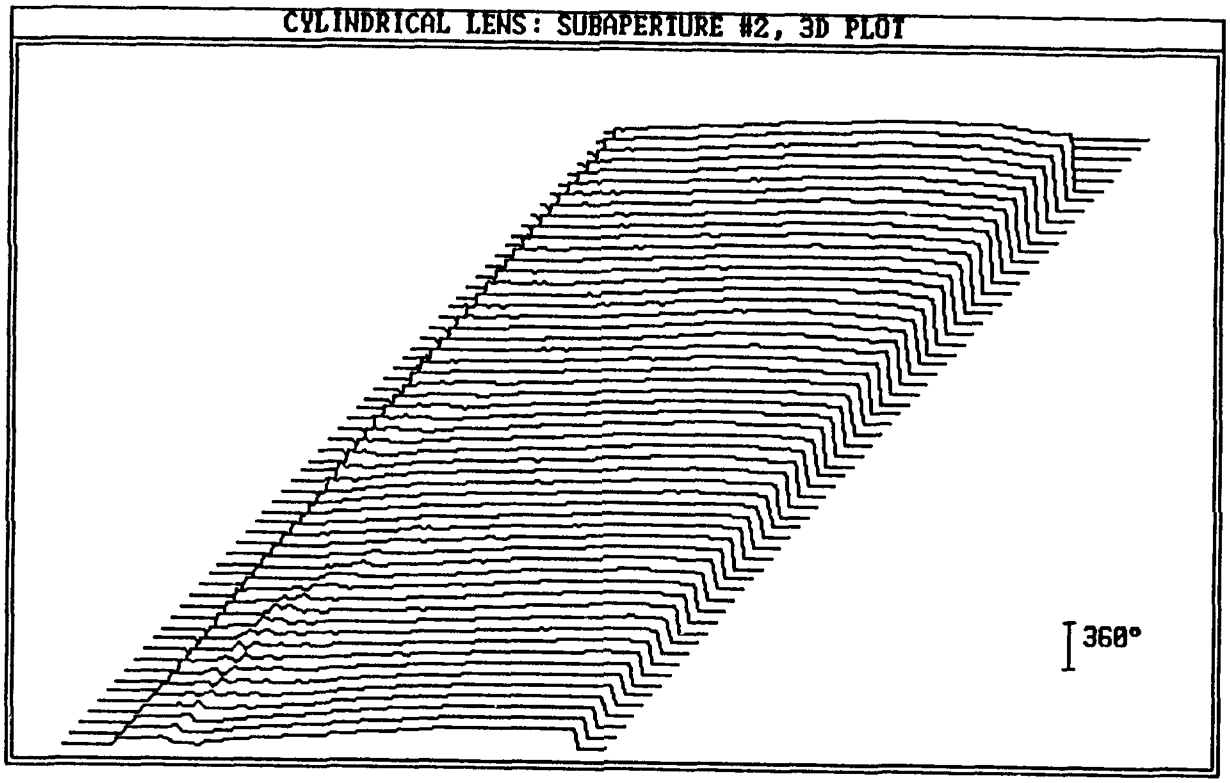
FIGURE 10: PHASE MEASUREMENT OF INDIVIDUAL SUBAPERTURES



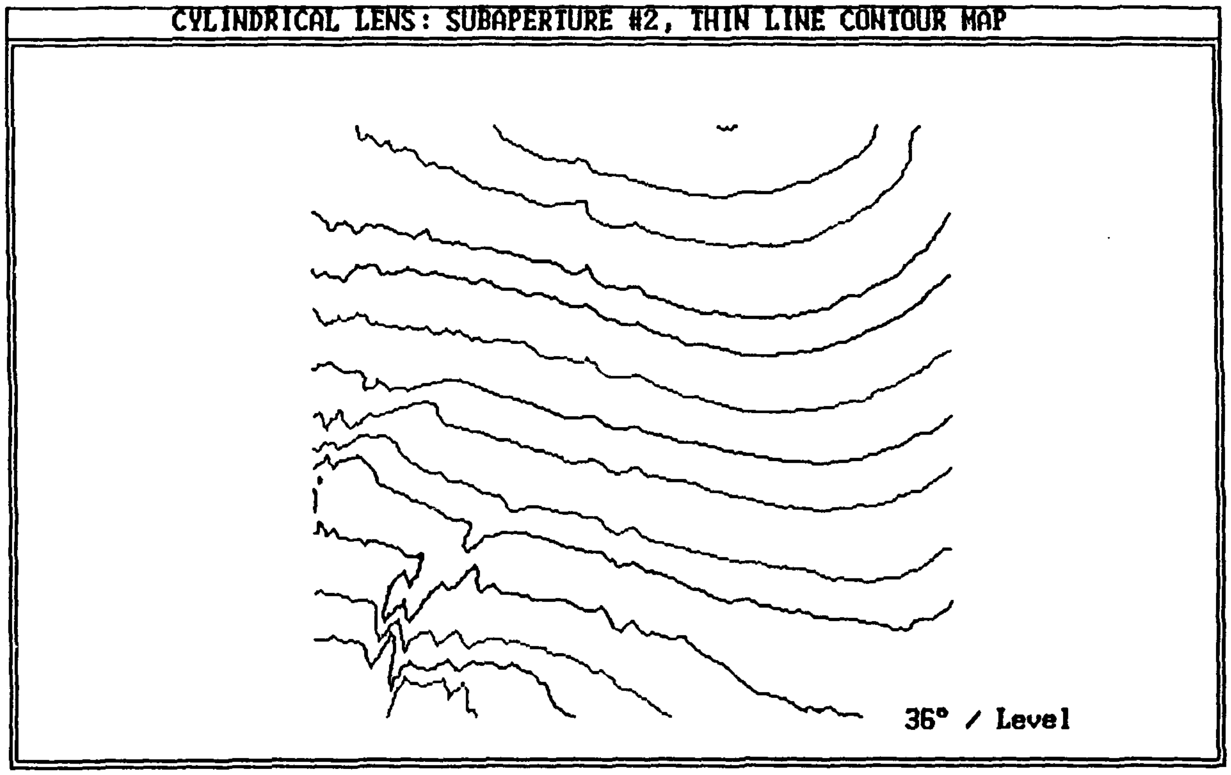
PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 1
(3-D ISOMETRIC PLOT)



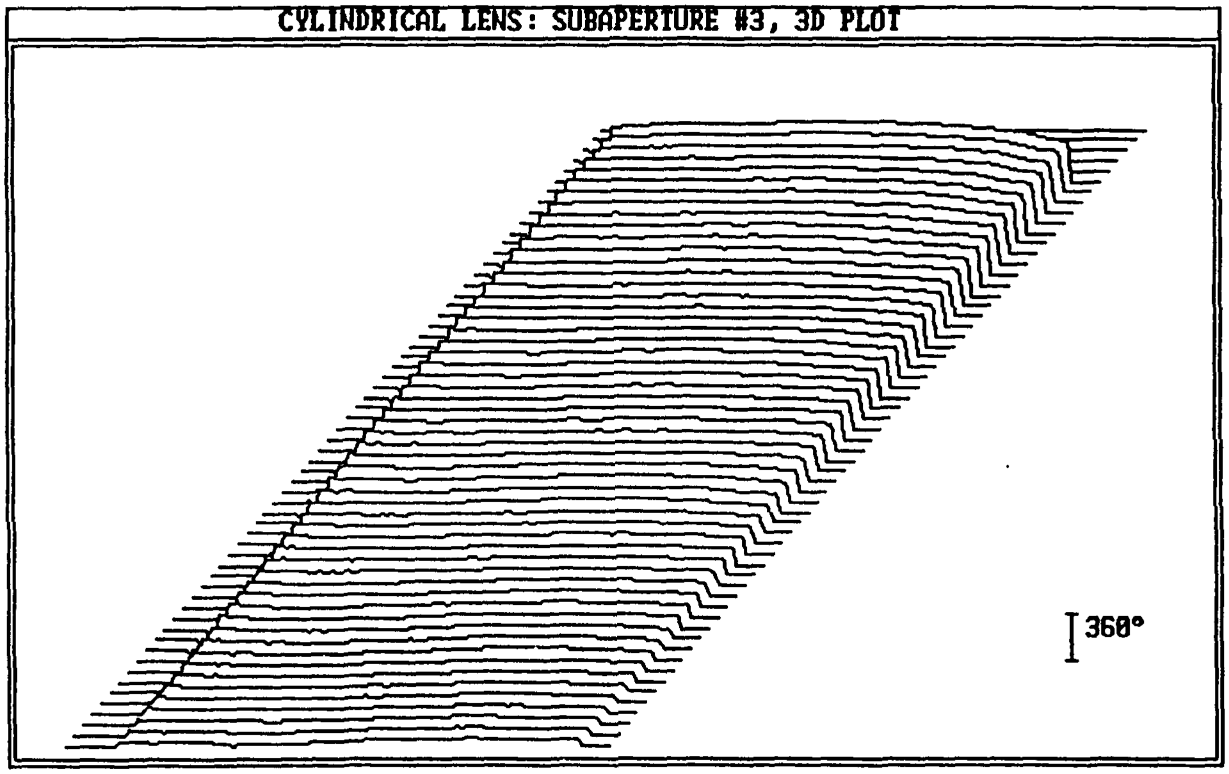
PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 1
(CONTOUR MAP)



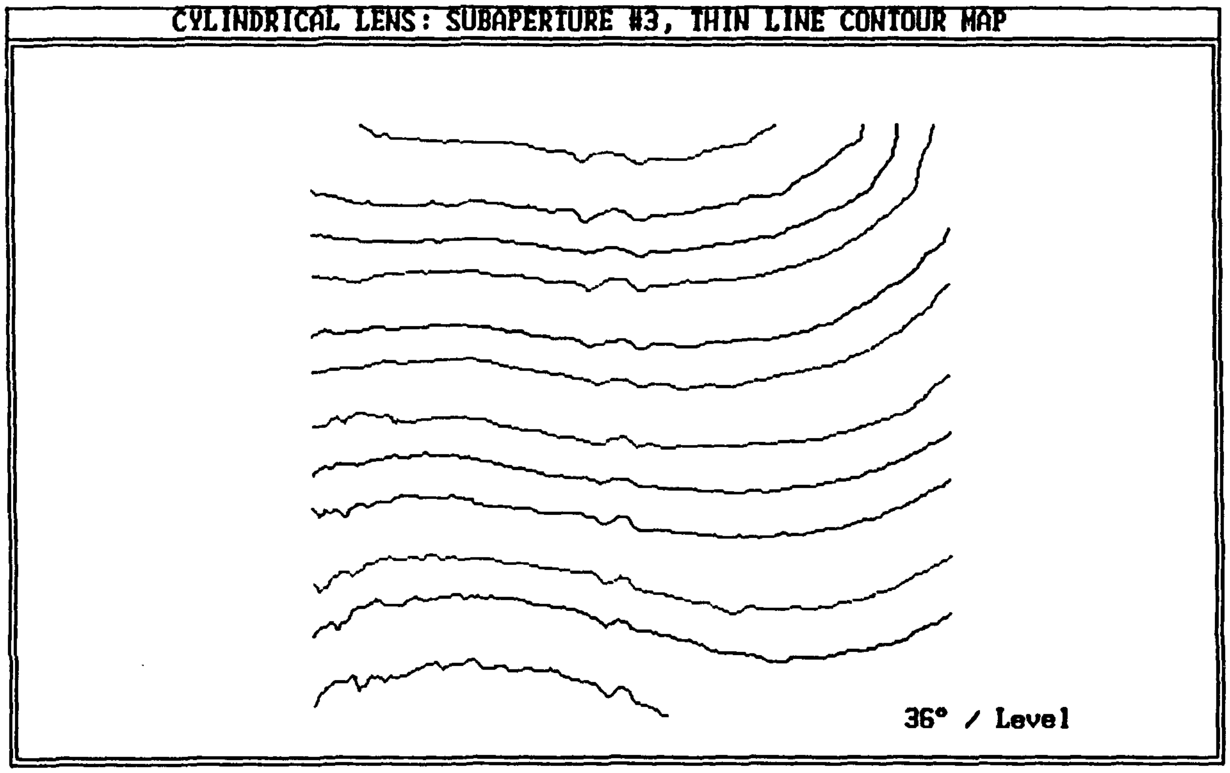
PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM #2
(3-D ISOMETRIC PLOT)



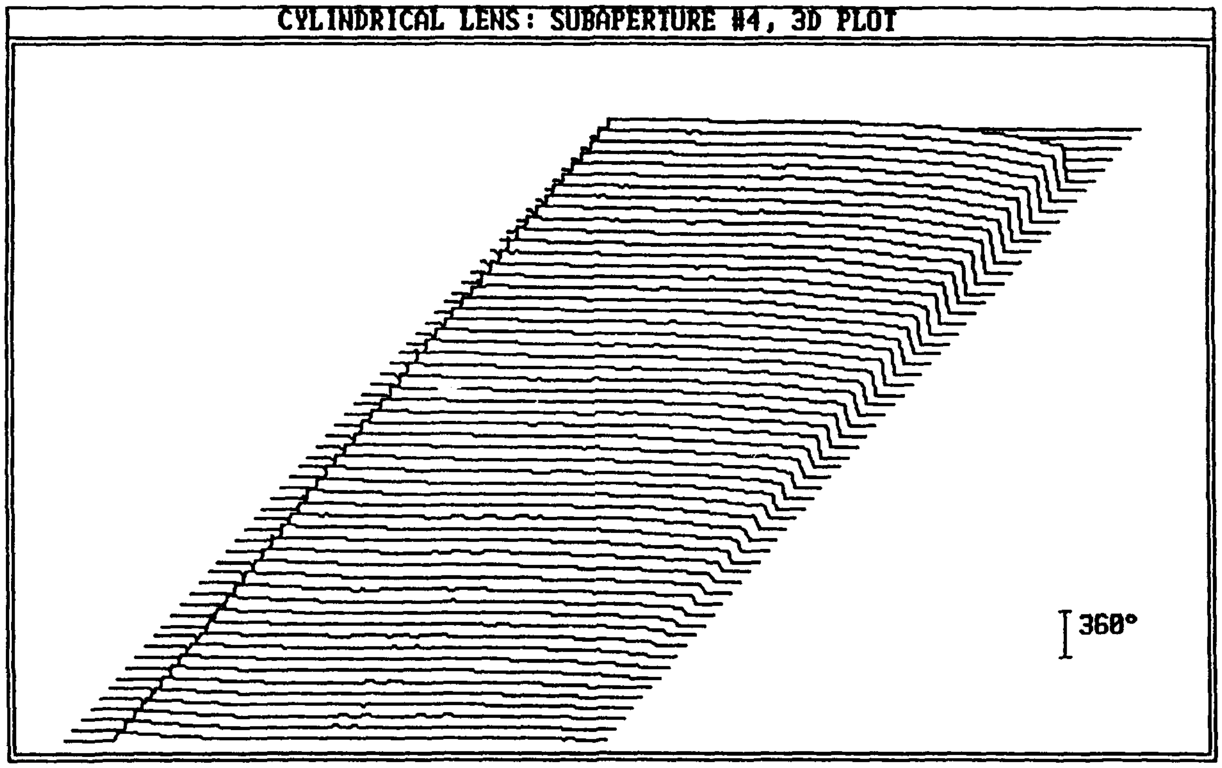
PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 2
(CONTOUR MAP)



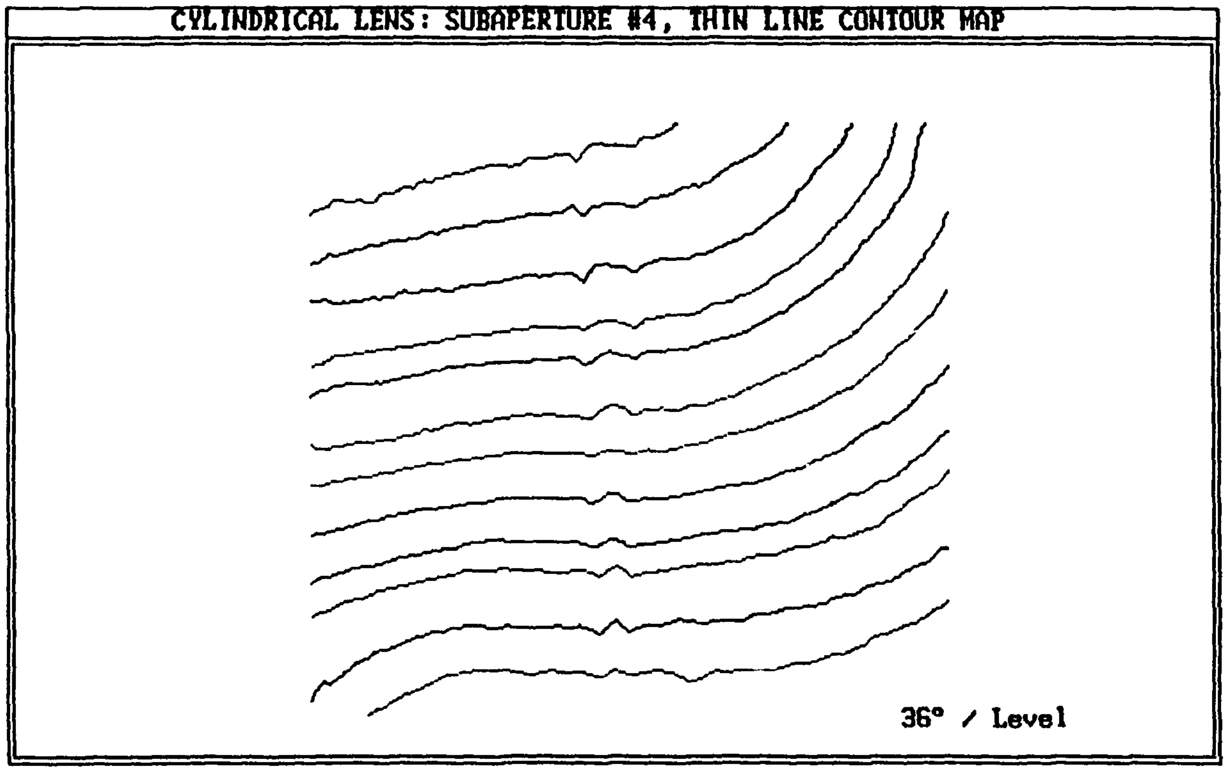
PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 3
(3-D ISOMETRIC PLOT)



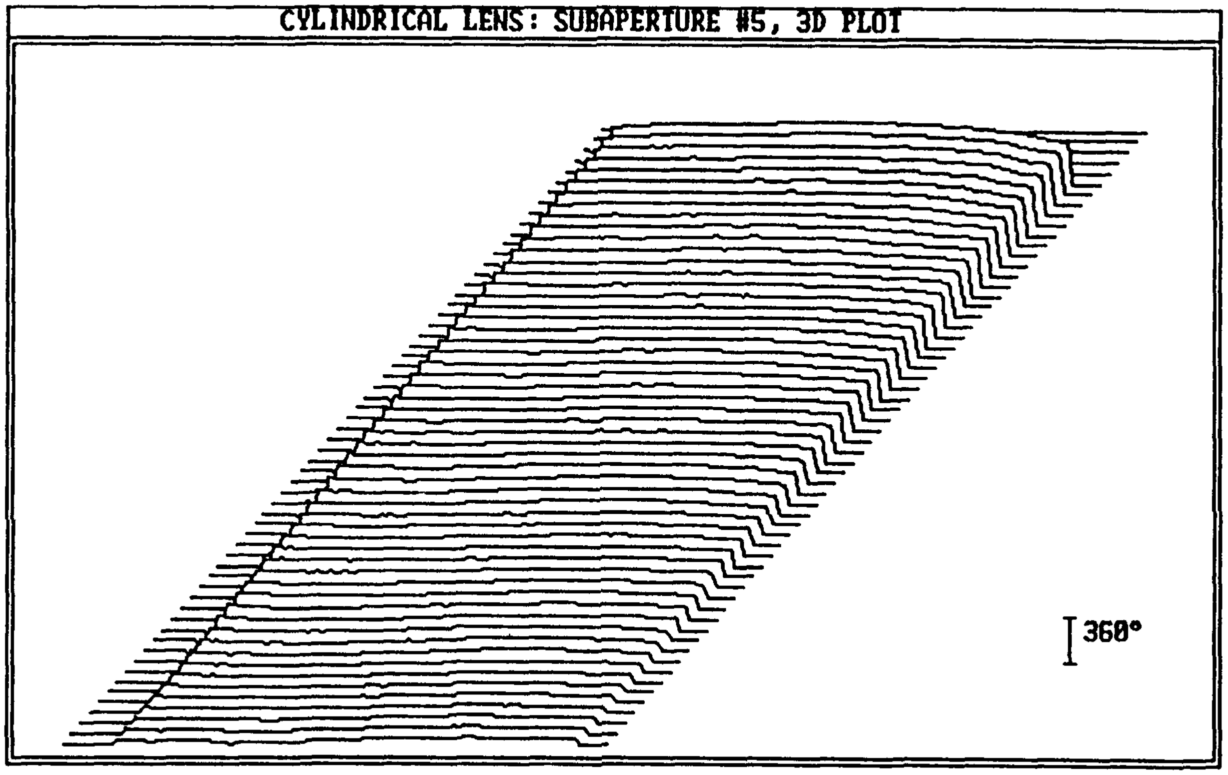
PHASE MEASUREMENT OF SUBAPERTURE INTERFERPGRAM #3
(CONTOUR MAP)



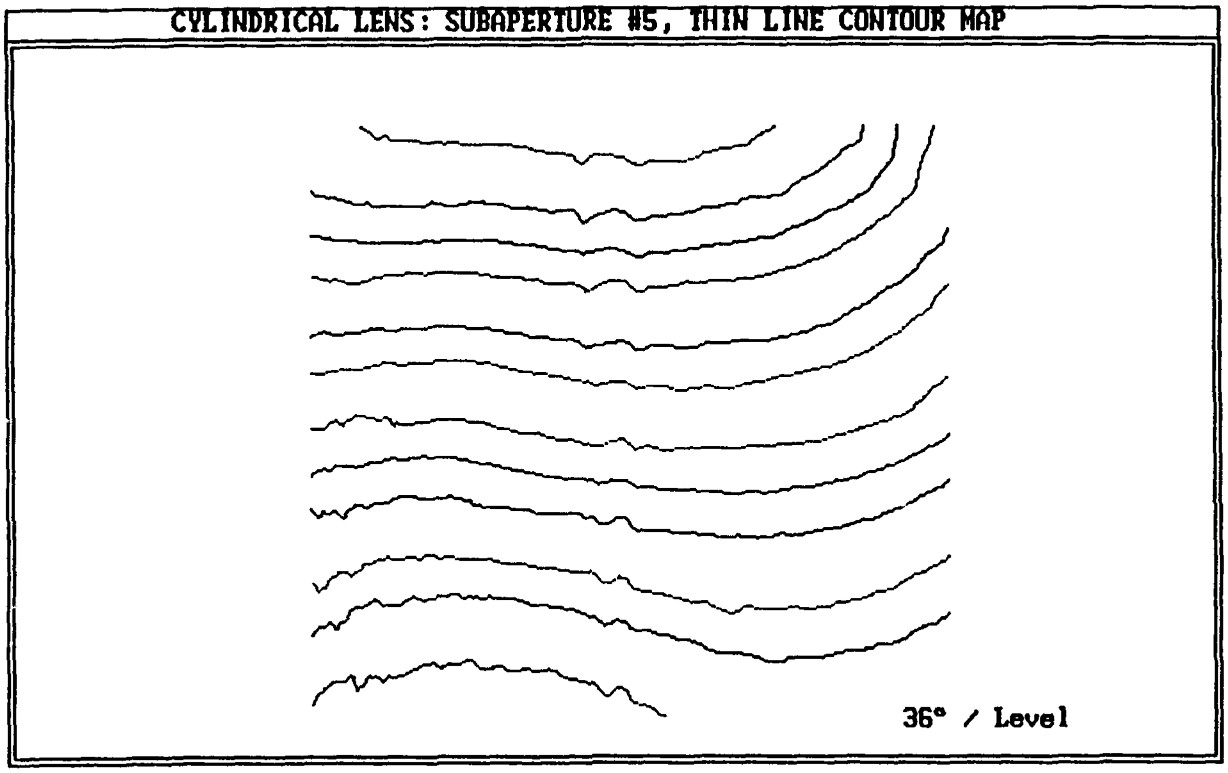
PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 4
(3-D ISOMETRIC PLOT)



PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 4
(CONTOUR MAP)

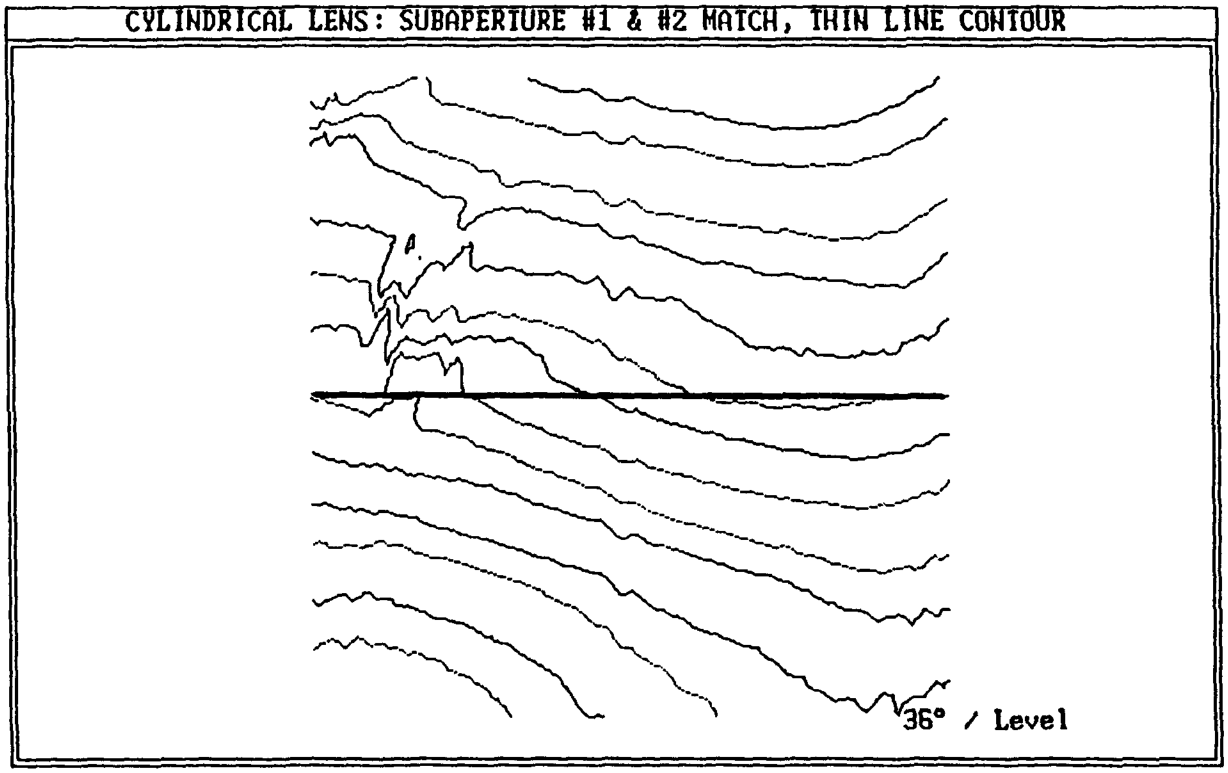


PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 5
(3-D ISOMETRIC PLOT)

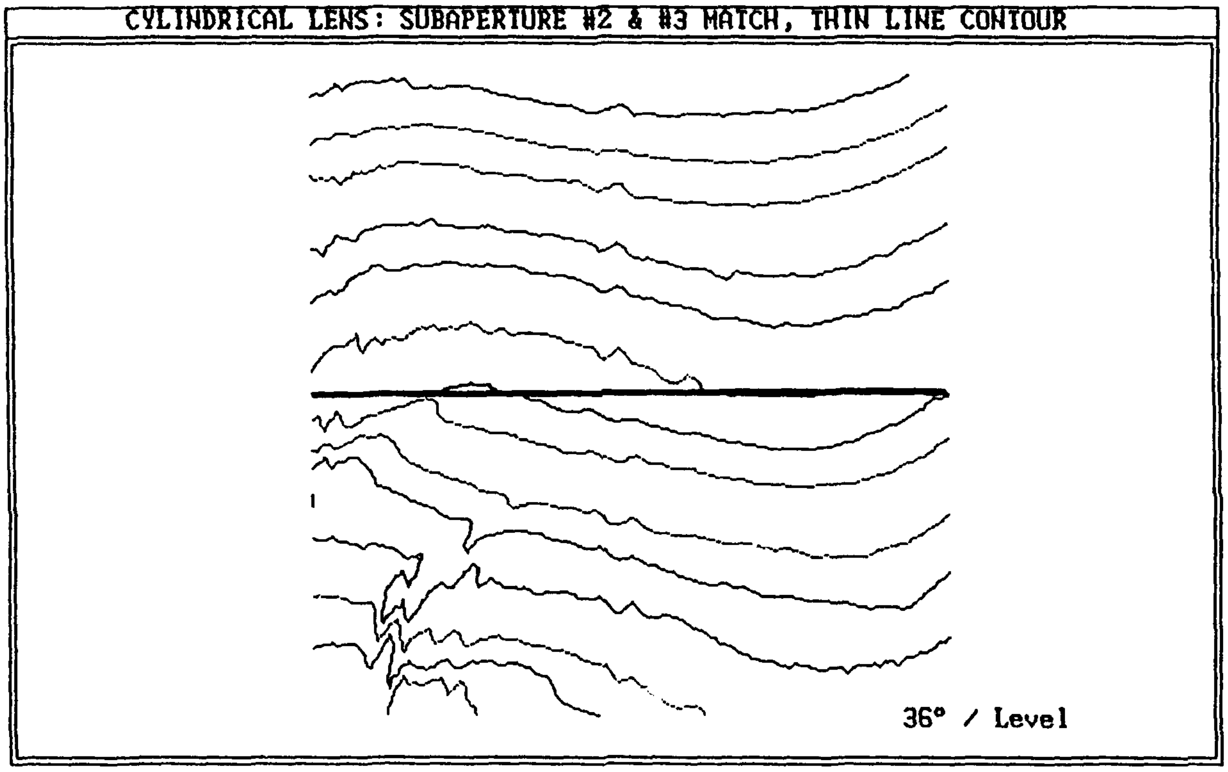


PHASE MEASUREMENT OF SUBAPERTURE INTERFEROGRAM # 5
(CONTOUR MAP)

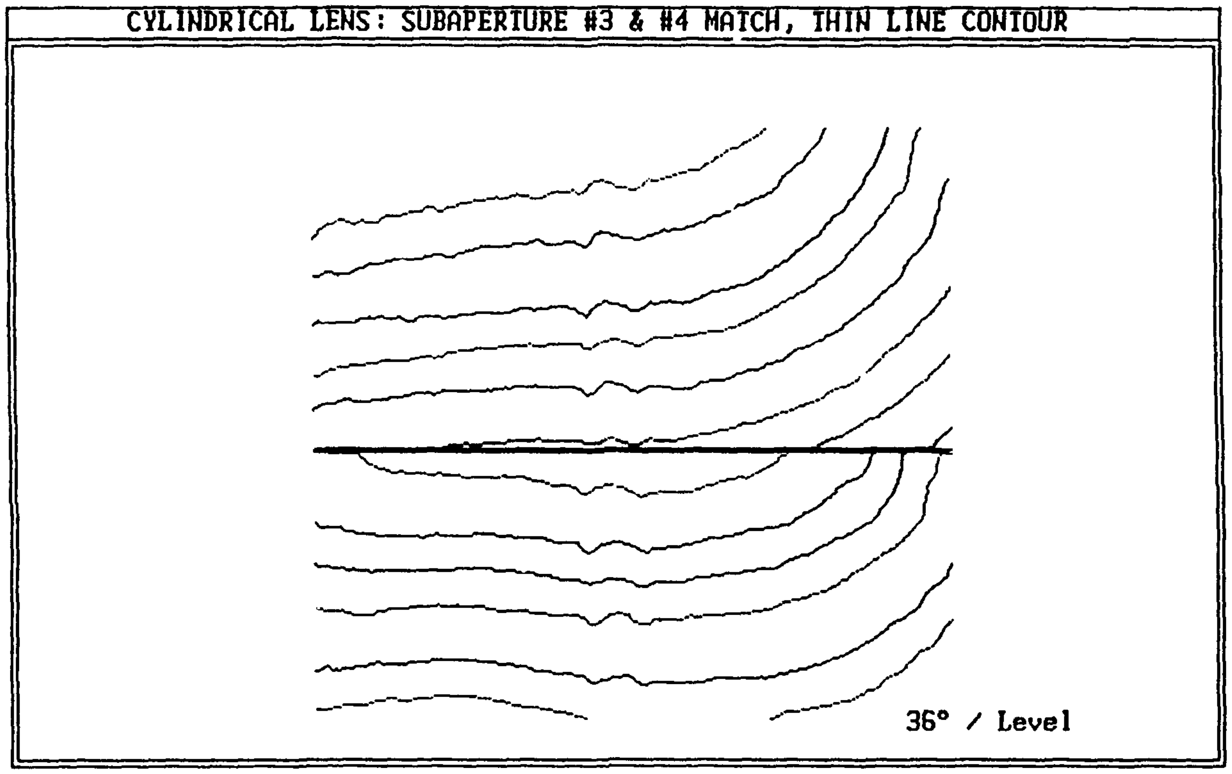
FIGURE 11: PHASE MEASUREMENT SHOWING LEAST SQUARE FITTING IN SUBAPERTURE OVERLAP AREAS



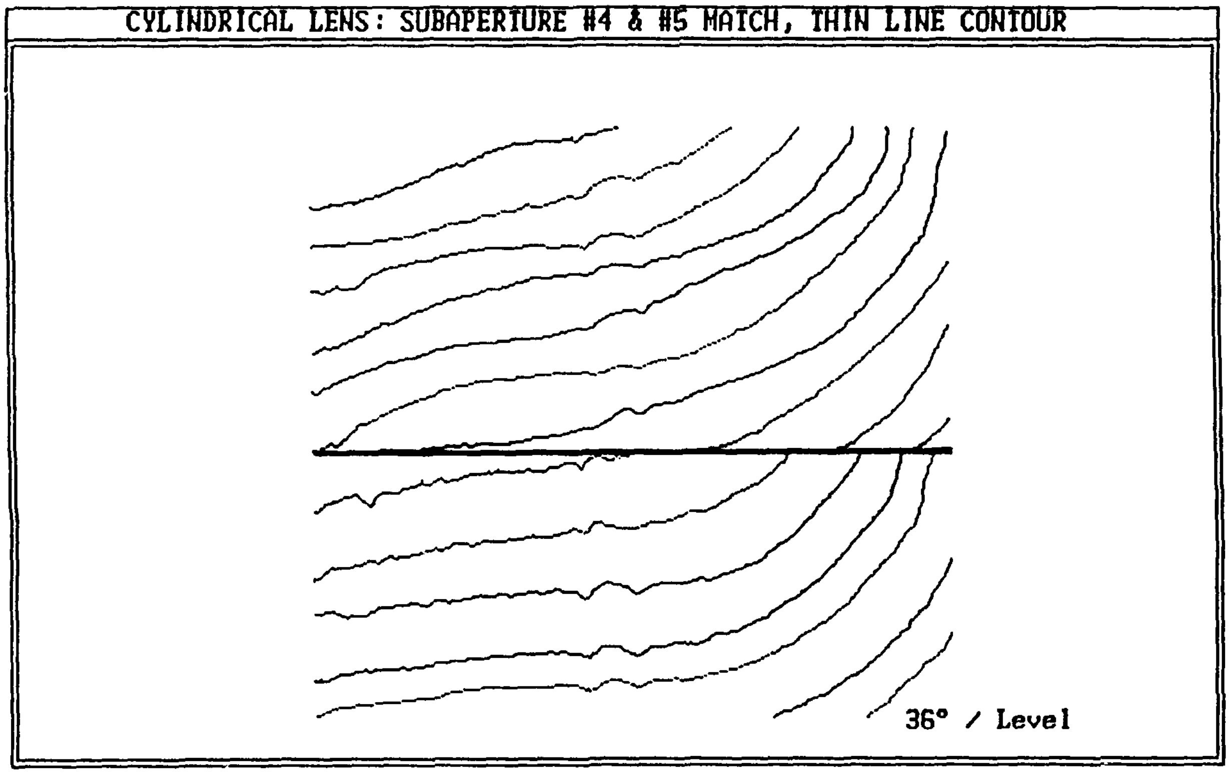
PHASE MEASUREMENT SHOWING LEAST SQUARE FITTING IN OVERLAP REGION FOR SUBAPERTURE INTERFEROGRAMS # 1 AND 2



PHASE MEASUREMENT SHOWING LEAST SQUARE FITTING IN
OVERLAP REGION FOR SUBAPERTURE INTERFEROGRAMS # 2 AND 3

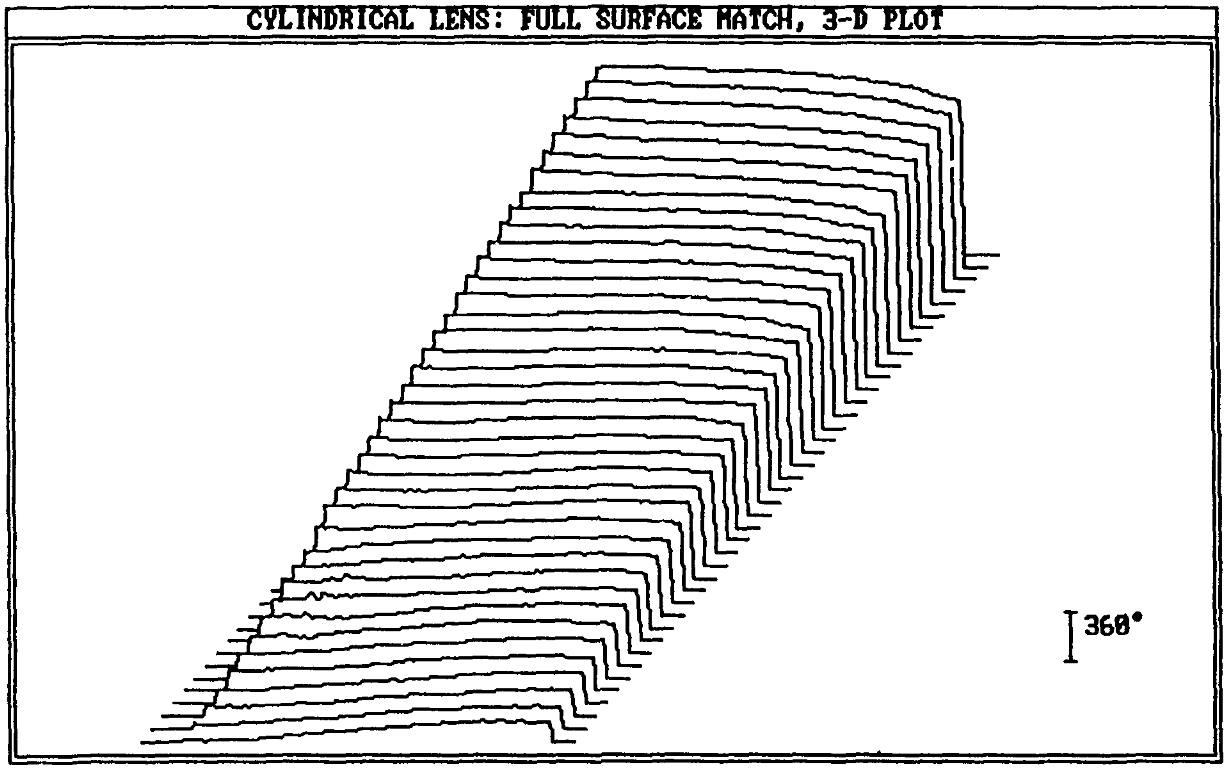


PHASE MEASUREMENT SHOWING LEAST SQUARE FITTING IN OVERLAP REGION FOR SUBAPERTURE INTERFEROGRAMS # 3 AND 4

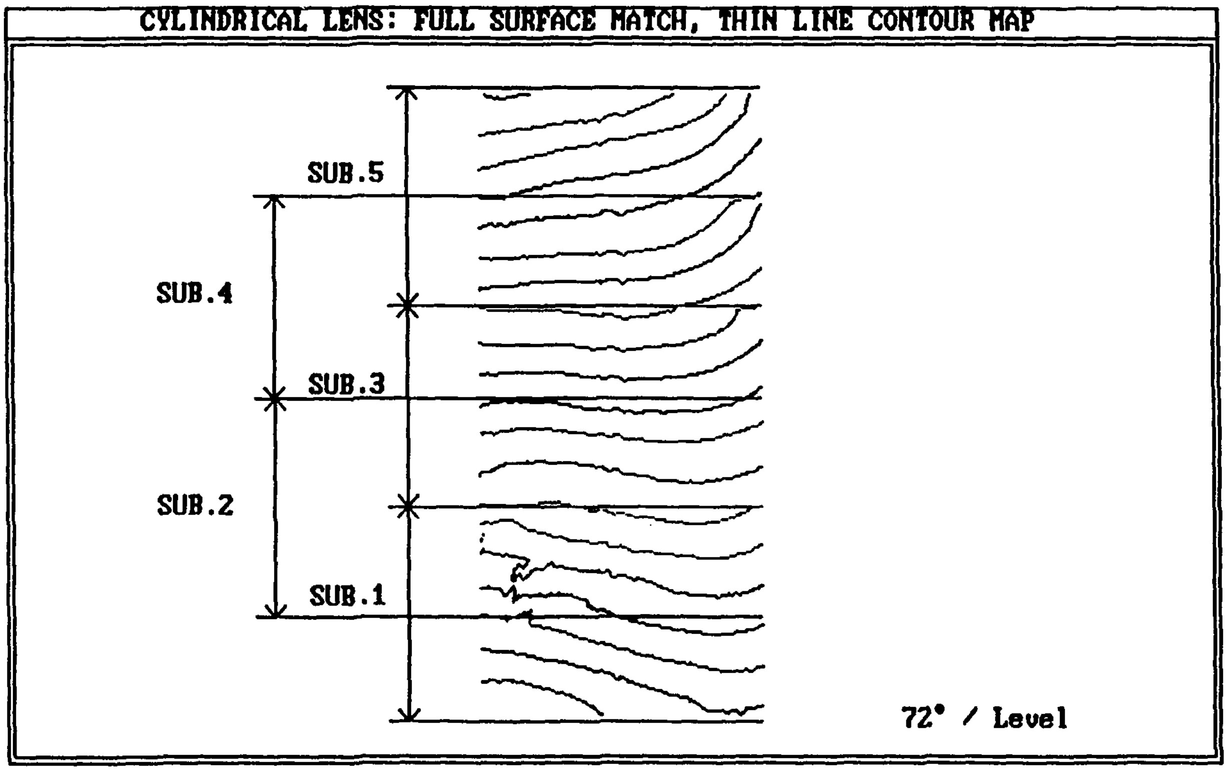


PHASE MEASUREMENT SHOWING LEAST SQUARE FITTING IN OVERLAP REGION FOR SUBAPERTURE INTERFEROGRAMS # 4 AND 5

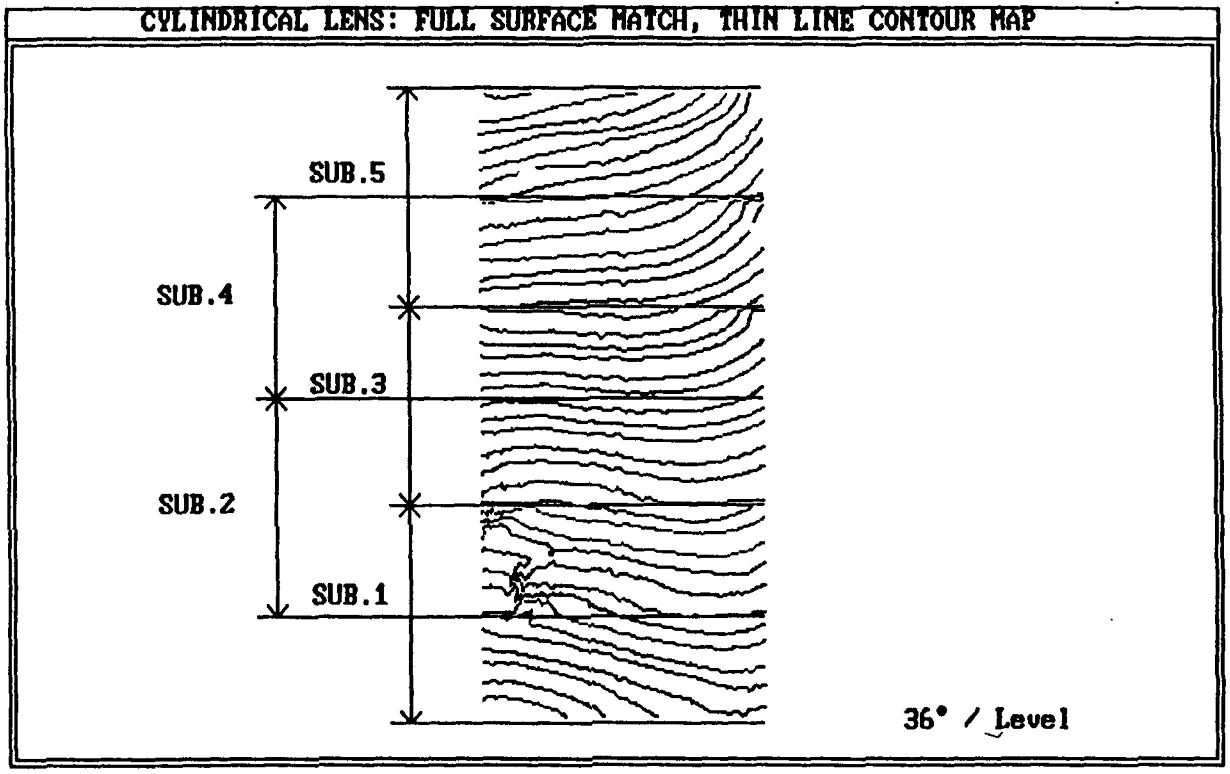
FIGURE 12: SYNTHESIS OF FULL SURFACE PHASE MEASUREMENT



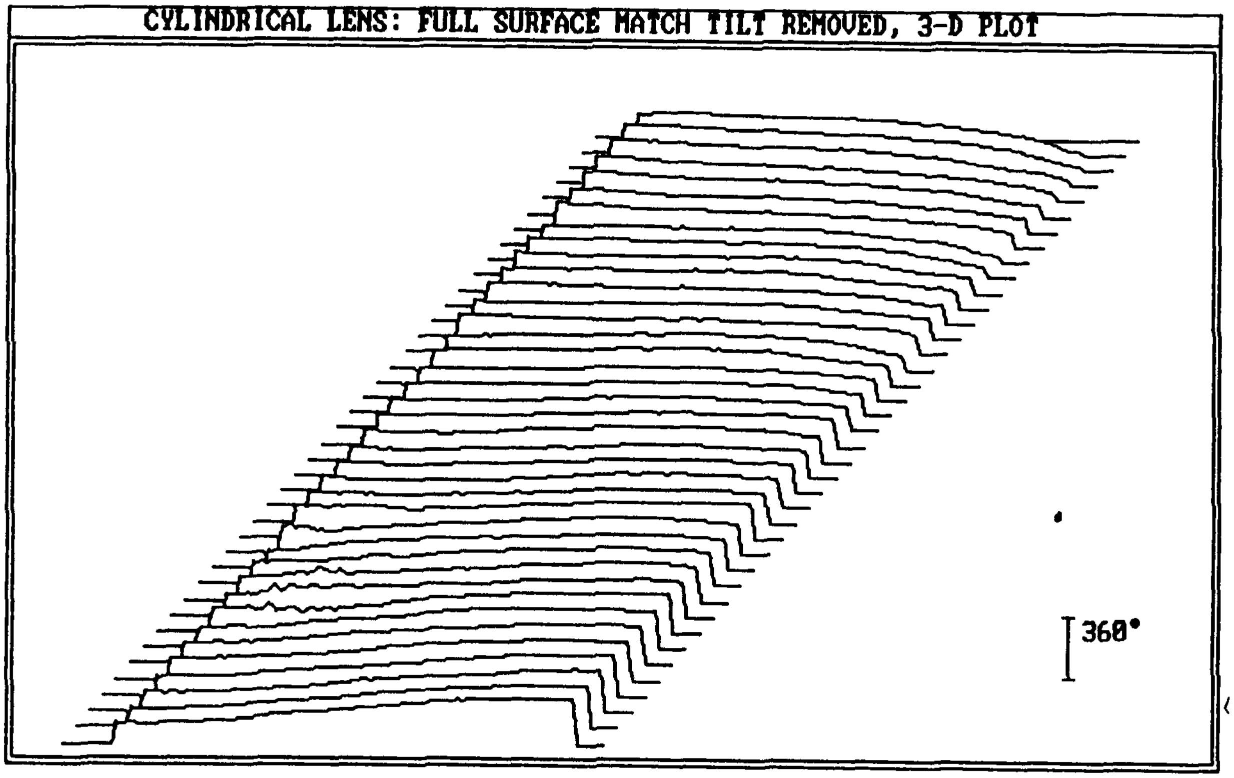
SYNTHESIS OF FULL-SURFACE PHASE MEASUREMENT FROM
5 SUBAPERTURE INTERFEROGRAMS (3-D ISOMETRIC PLOT)



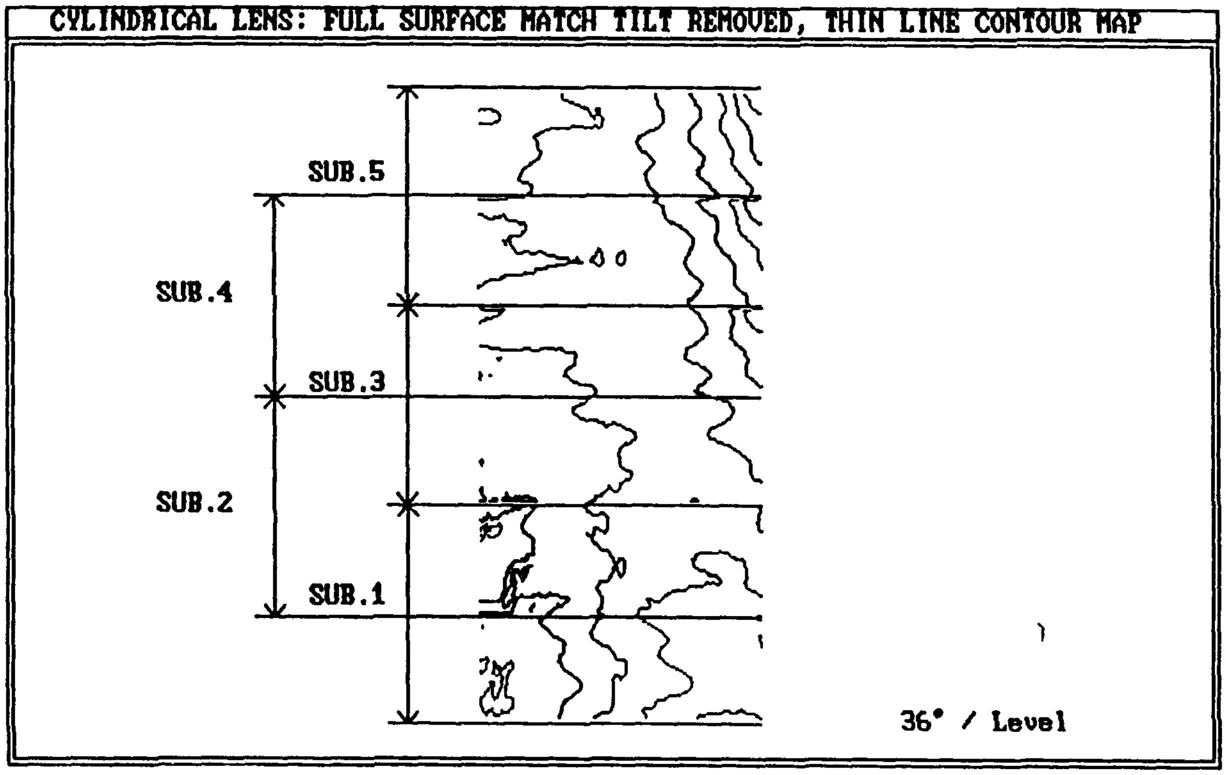
SYNTHESIS OF FULL-SURFACE PHASE MEASUREMENT FROM 5 SUBAPERTURE INTERFEROGRAMS (CONTOUR MAP)



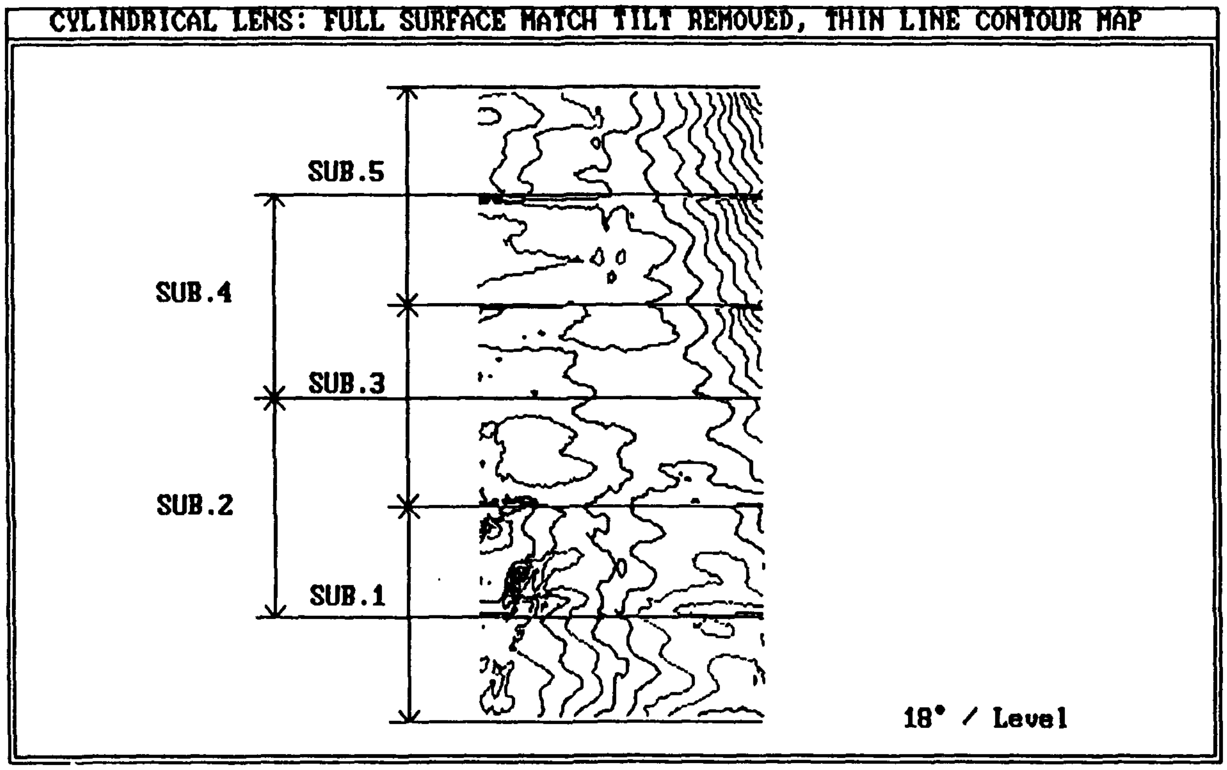
SYNTHESIS OF FULL-SURFACE PHASE MEASUREMENT FROM 5 SUBAPERTURE INTERFEROGRAMS (CONTOUR MAP)



SYNTHESIS OF FULL-SURFACE PHASE MEASUREMENT FROM 5 SUBAPERTURE INTERFEROGRAMS WITH TILT REMOVED (3-D ISOMETRIC PLOT)

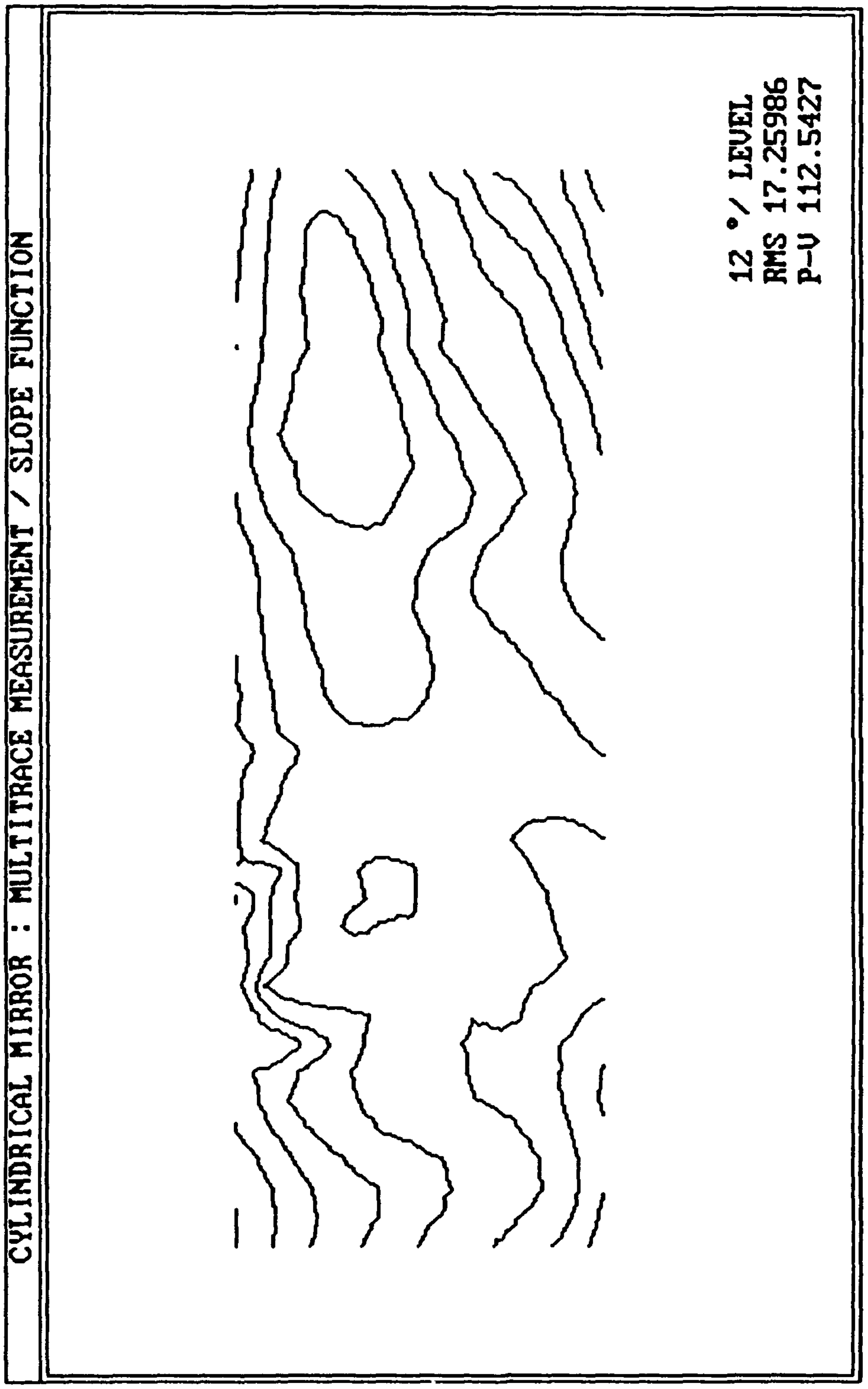


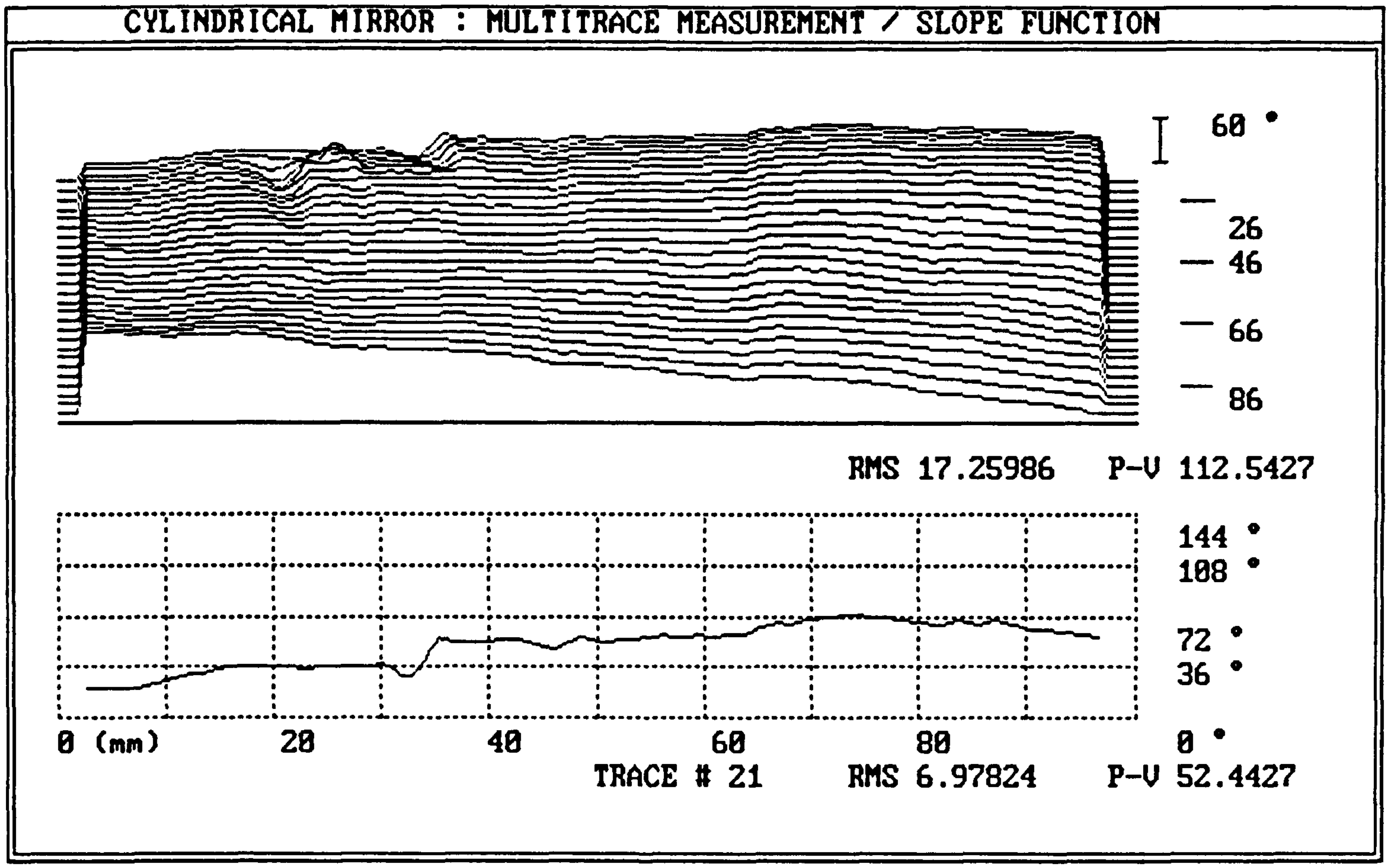
SYNTHESIS OF FULL-SURFACE PHASE MEASUREMENT FROM 5 SUBAPERTURE INTERFEROGRAMS WITH TILT REMOVED (CONTOUR MAP)

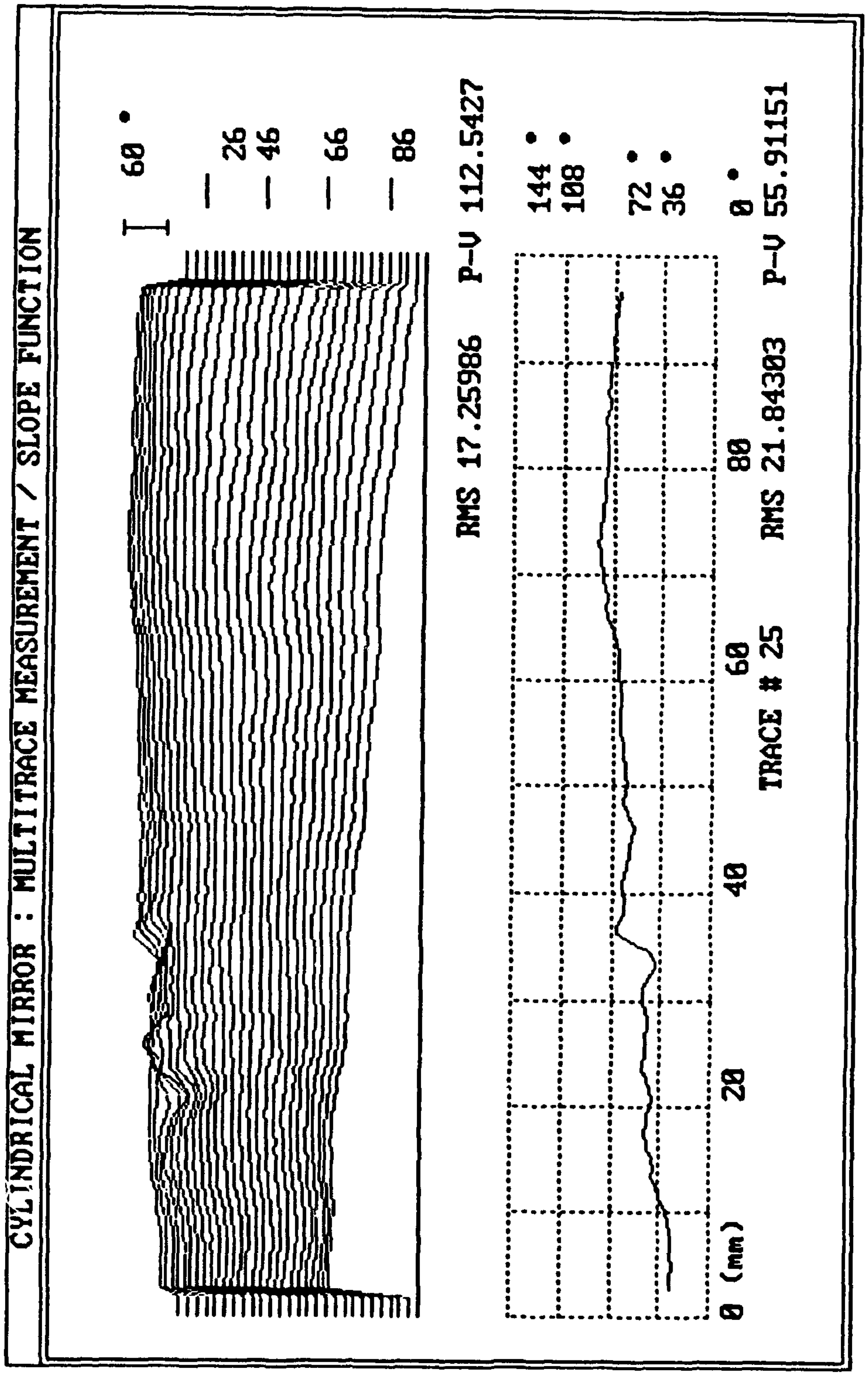


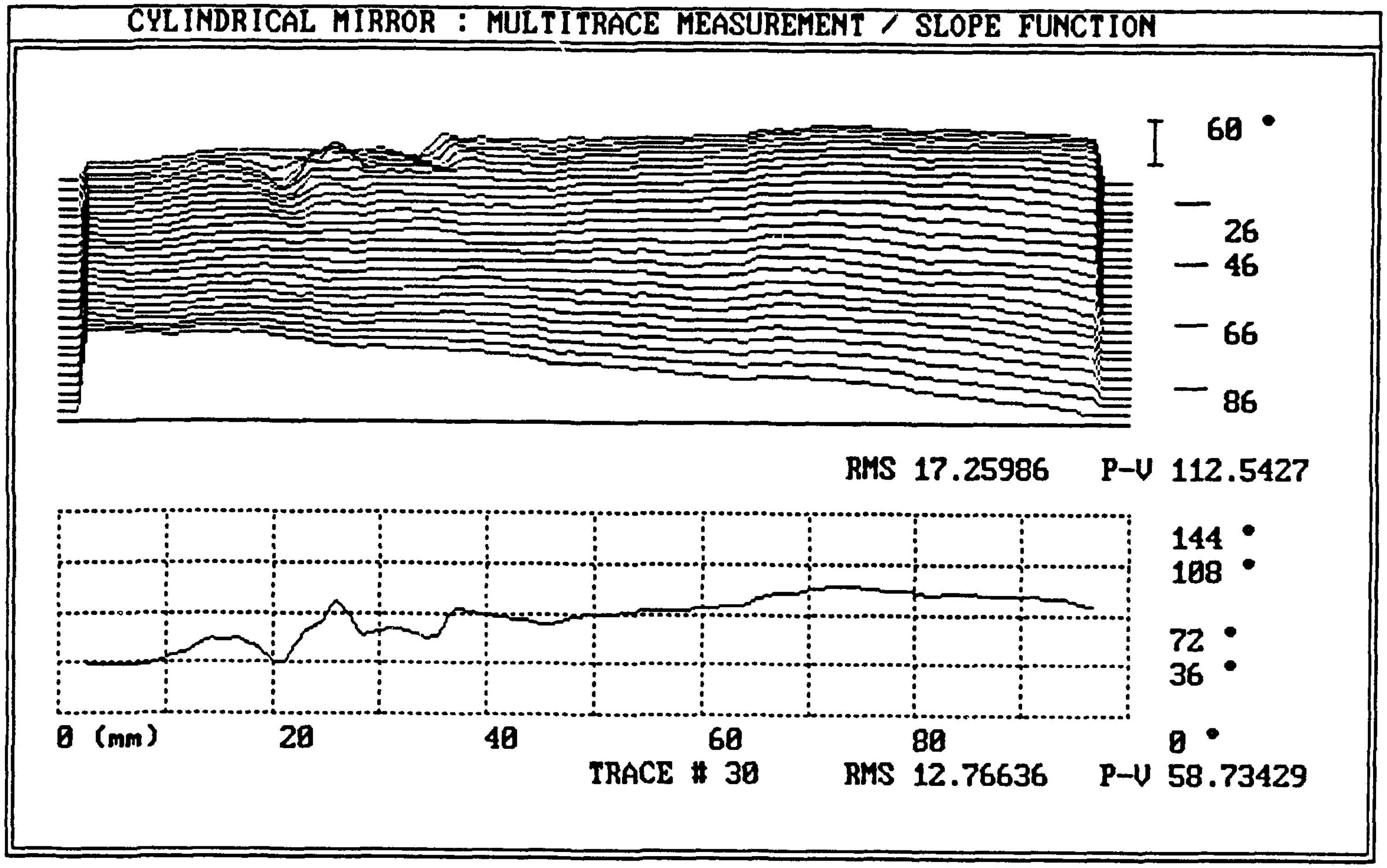
SYNTHESIS OF FULL-SURFACE PHASE MEASUREMENT FROM 5 SUBAPERTURE INTERFEROGRAMS WITH TILT REMOVED (CONTOUR MAP)

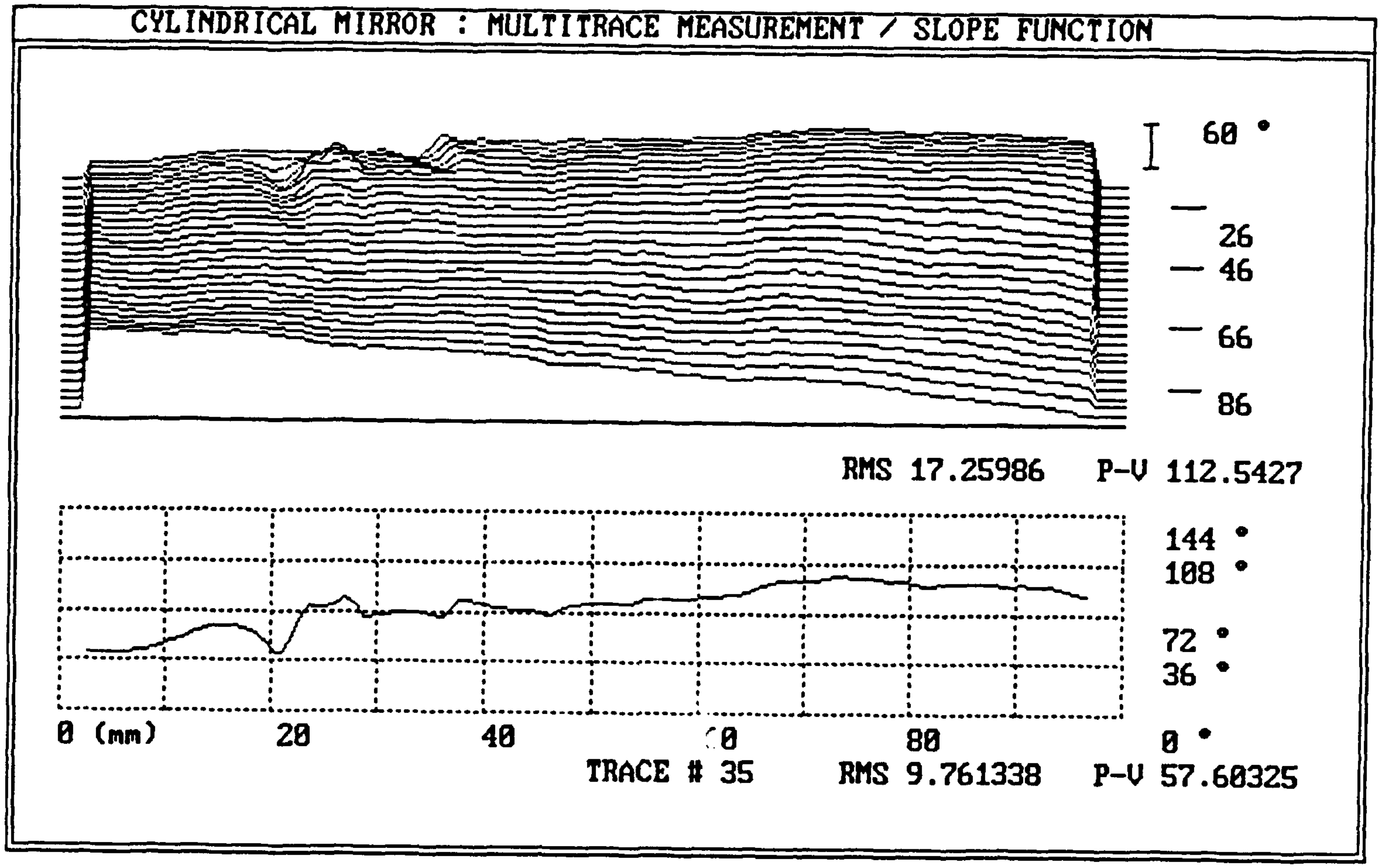
FIGURE 13: RESULTS OF SUBAPERTURE TESTING (SLOPE FUNCTION)

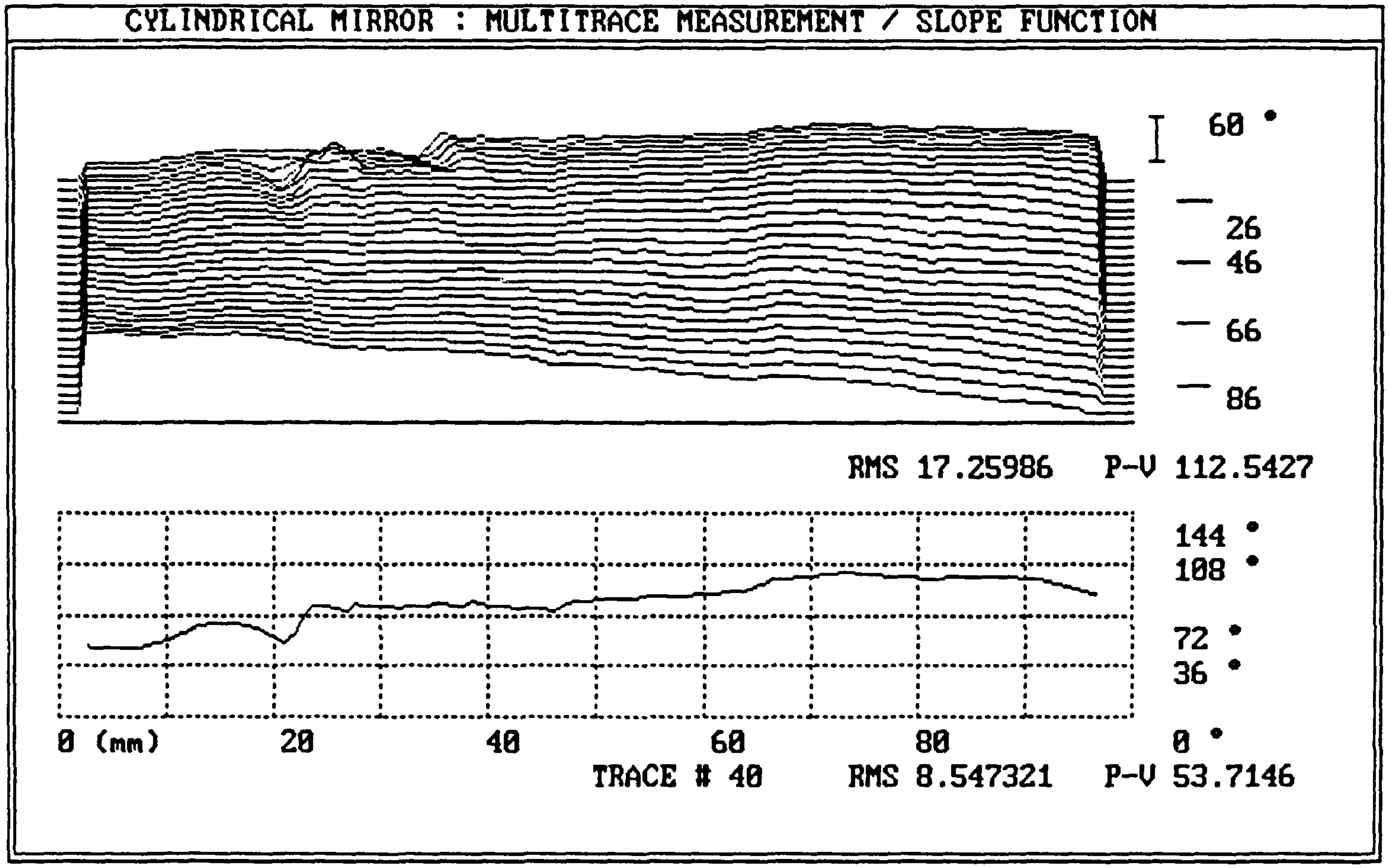

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
12 °/ LEVEL
RMS 17.25986
P-V 112.5427

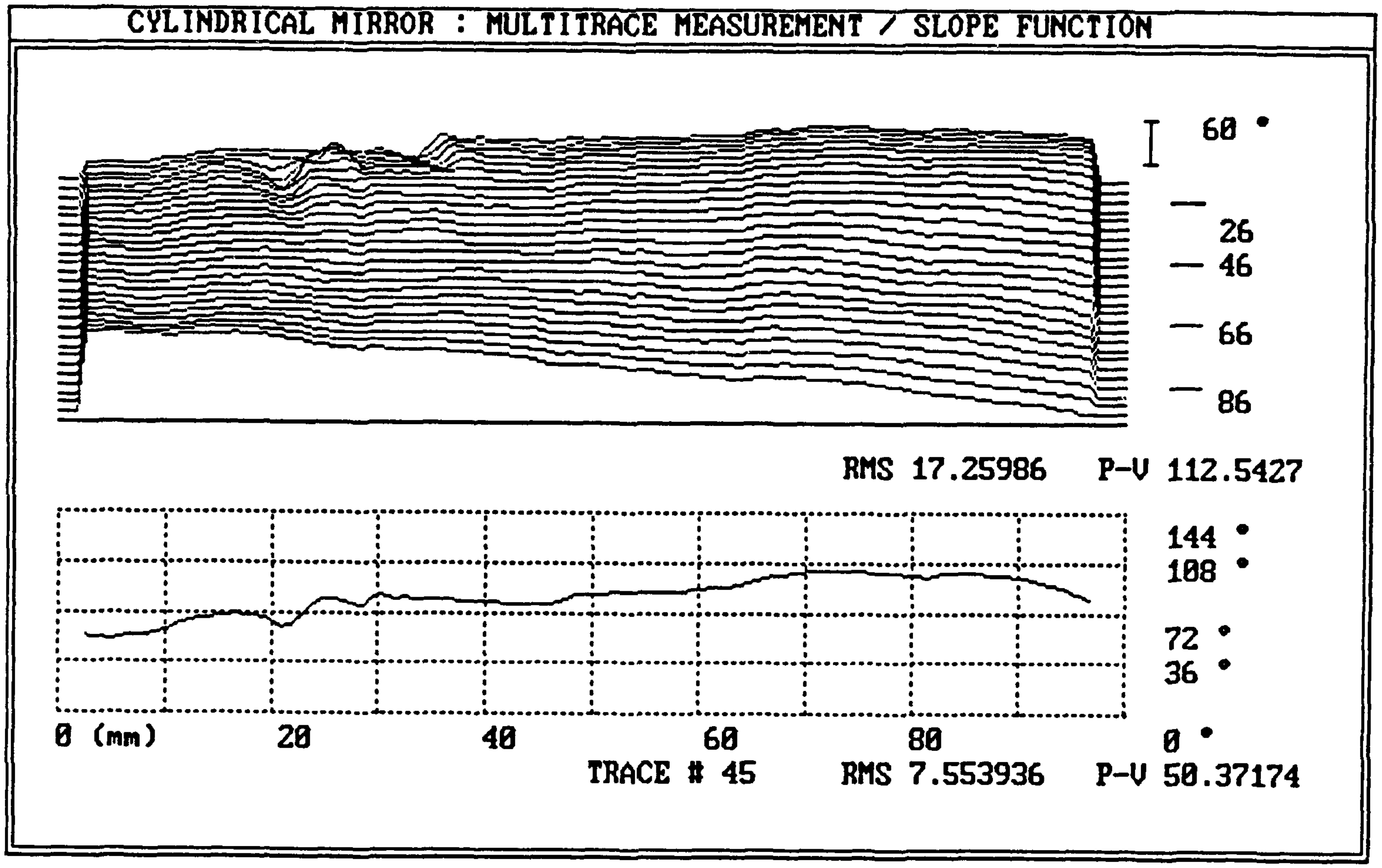

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986
P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 21
RMS 6.97824
P-V 52.4427

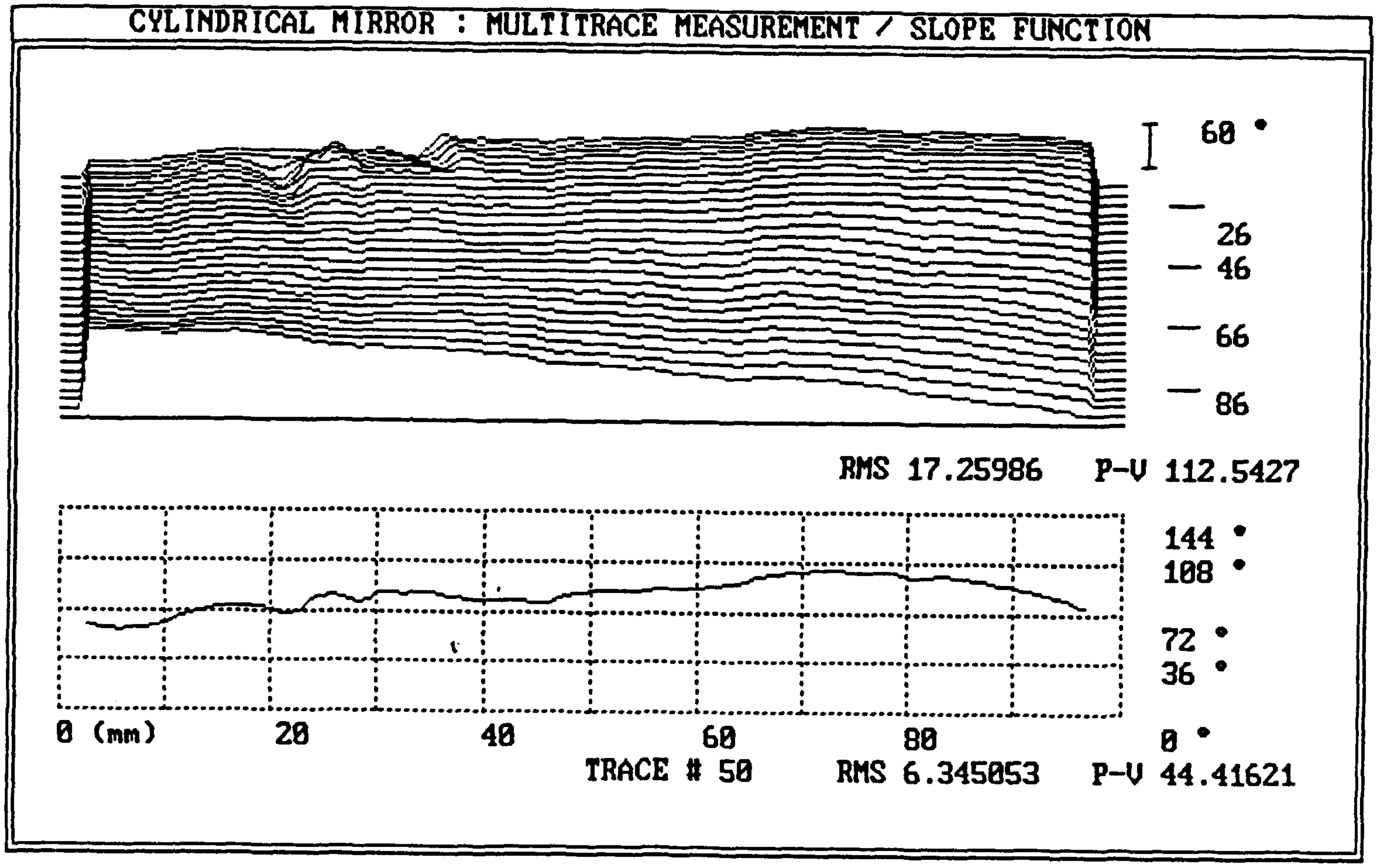

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986 P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 25
RMS 21.84303 P-V 55.91151

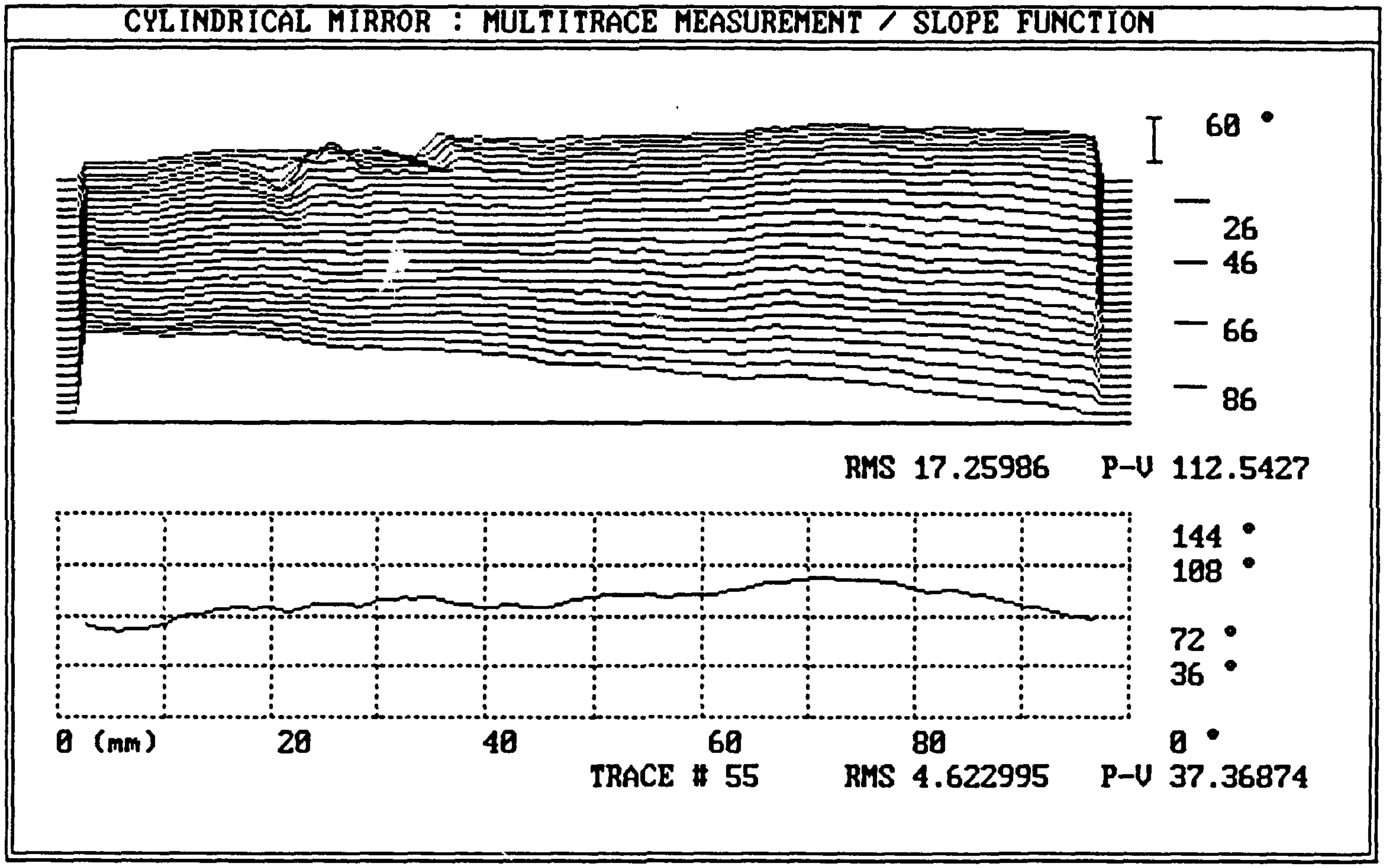

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986 P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 30
RMS 12.76636 P-V 58.73429


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 °
26
46
66
86
RMS 17.25986
P-V 112.5427
144 °
108 °
72 °
36 °
0 °
0 (mm)
20
40
60
80
TRACE # 35
RMS 9.761338
P-V 57.60325


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986
P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 40
RMS 8.547321
P-V 53.7146

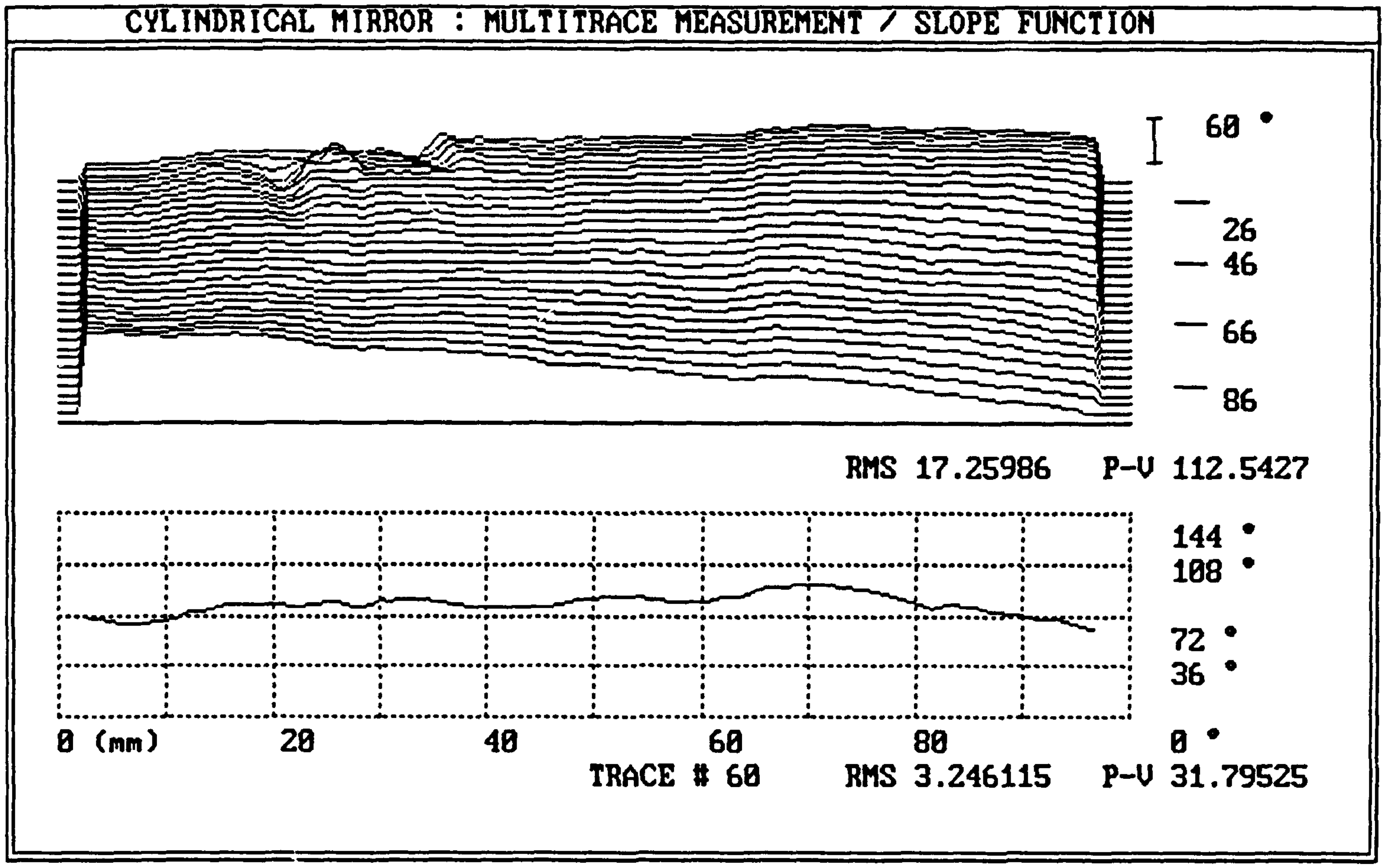

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986 P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 45
RMS 7.553936 P-V 50.37174

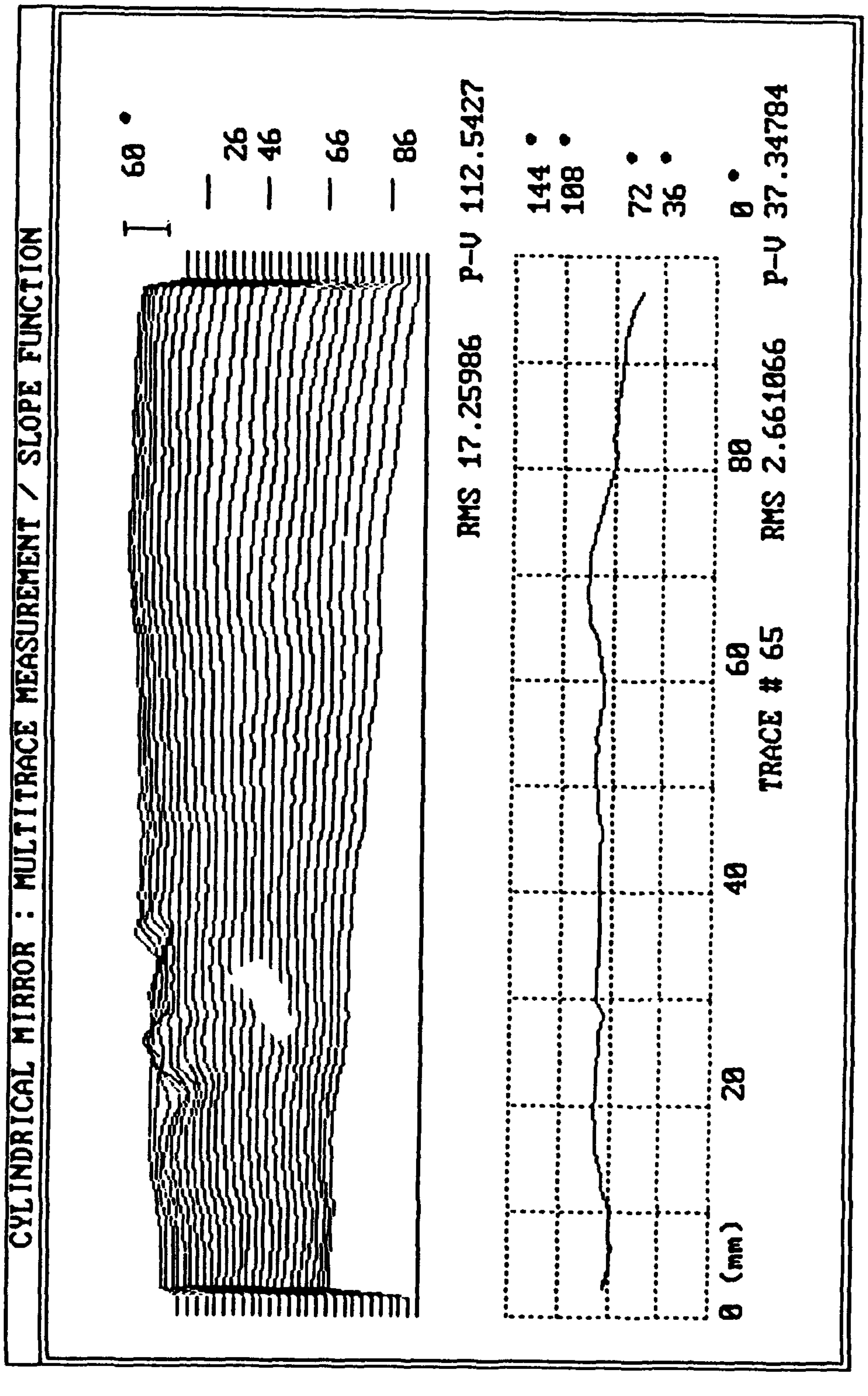

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 °
26
46
66
86
RMS 17.25986 P-V 112.5427
144 °
108 °
72 °
36 °
0 °
0 (mm)
20
40
60
80
TRACE # 50 RMS 6.345053 P-V 44.41621

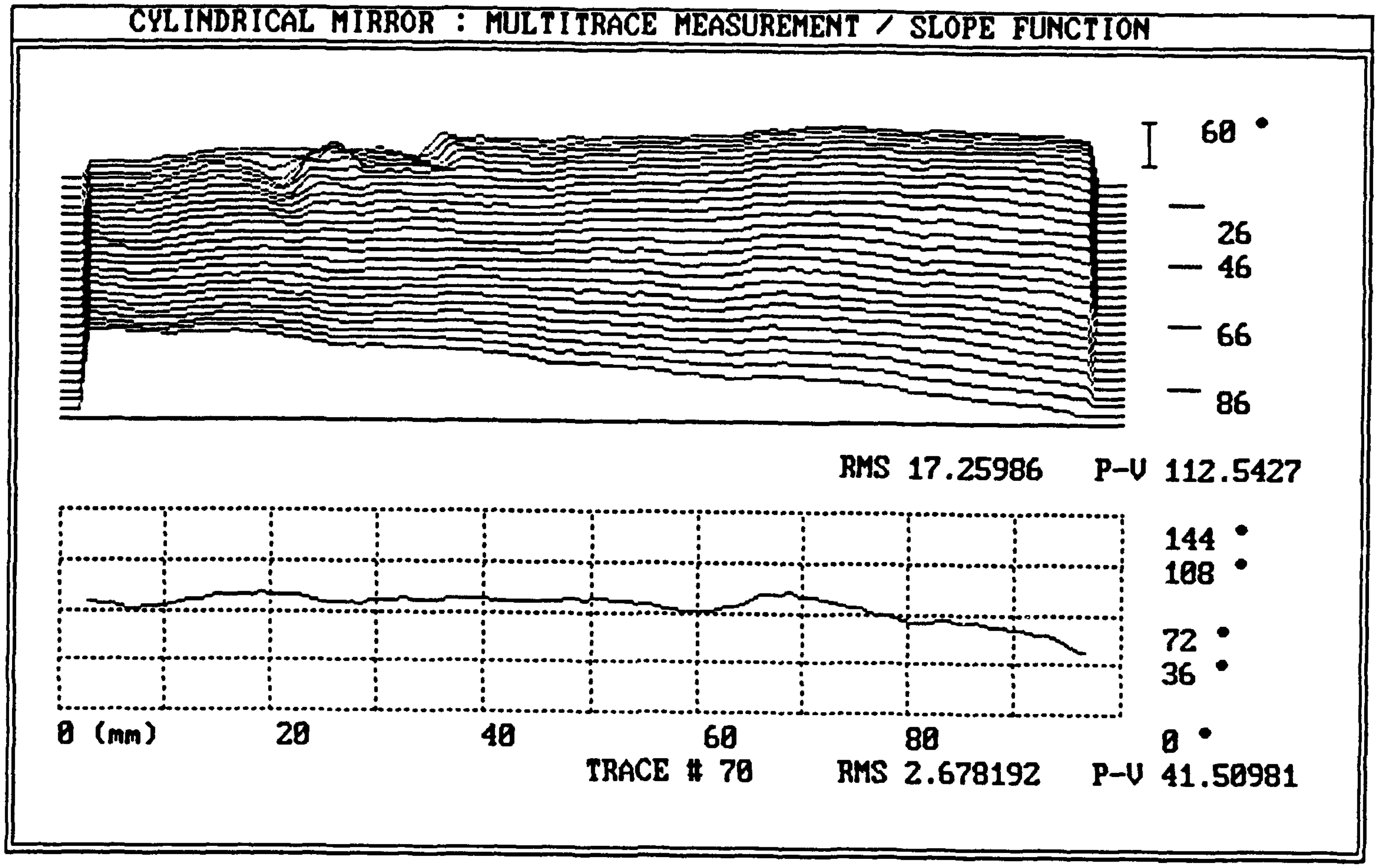


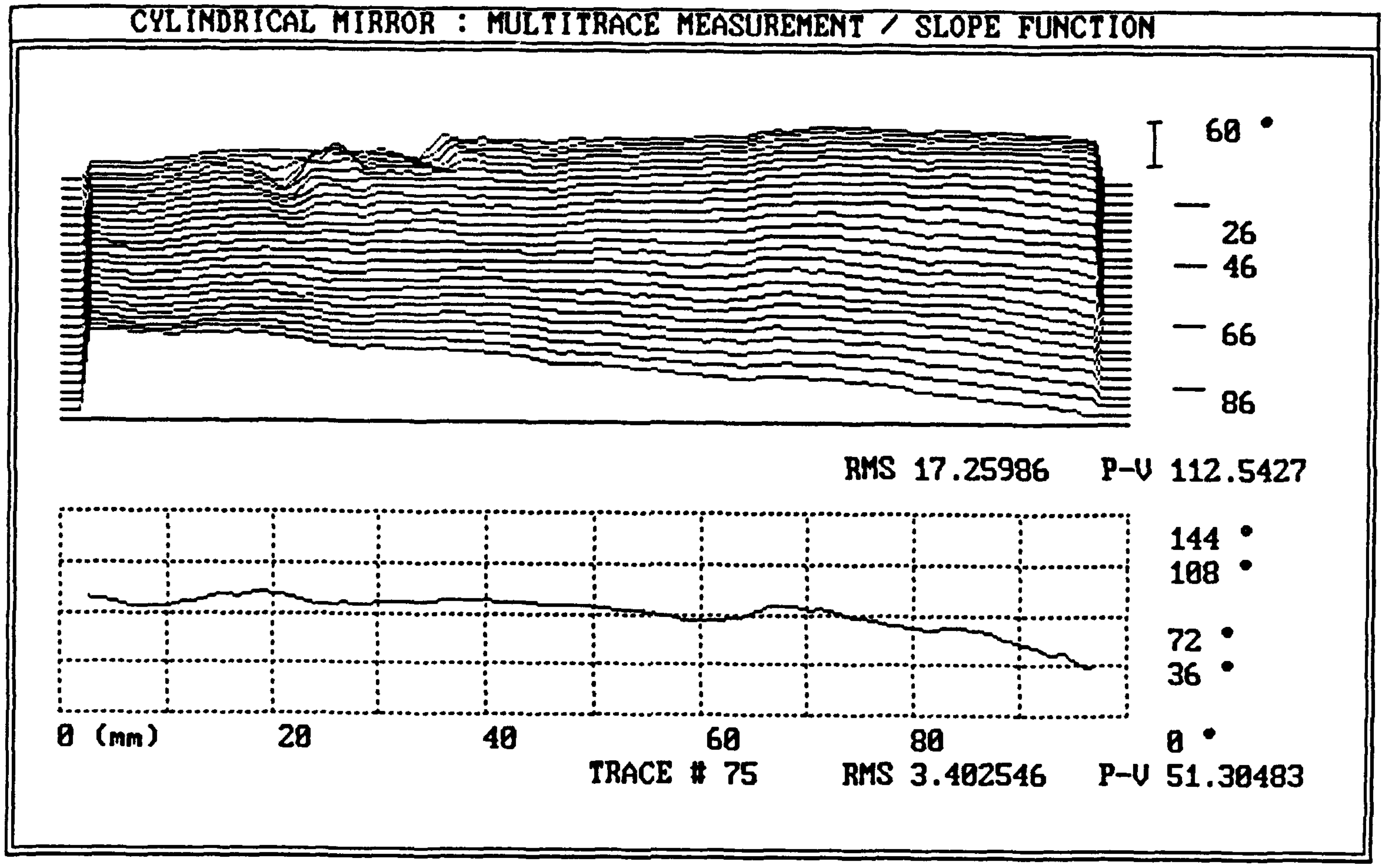
CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION

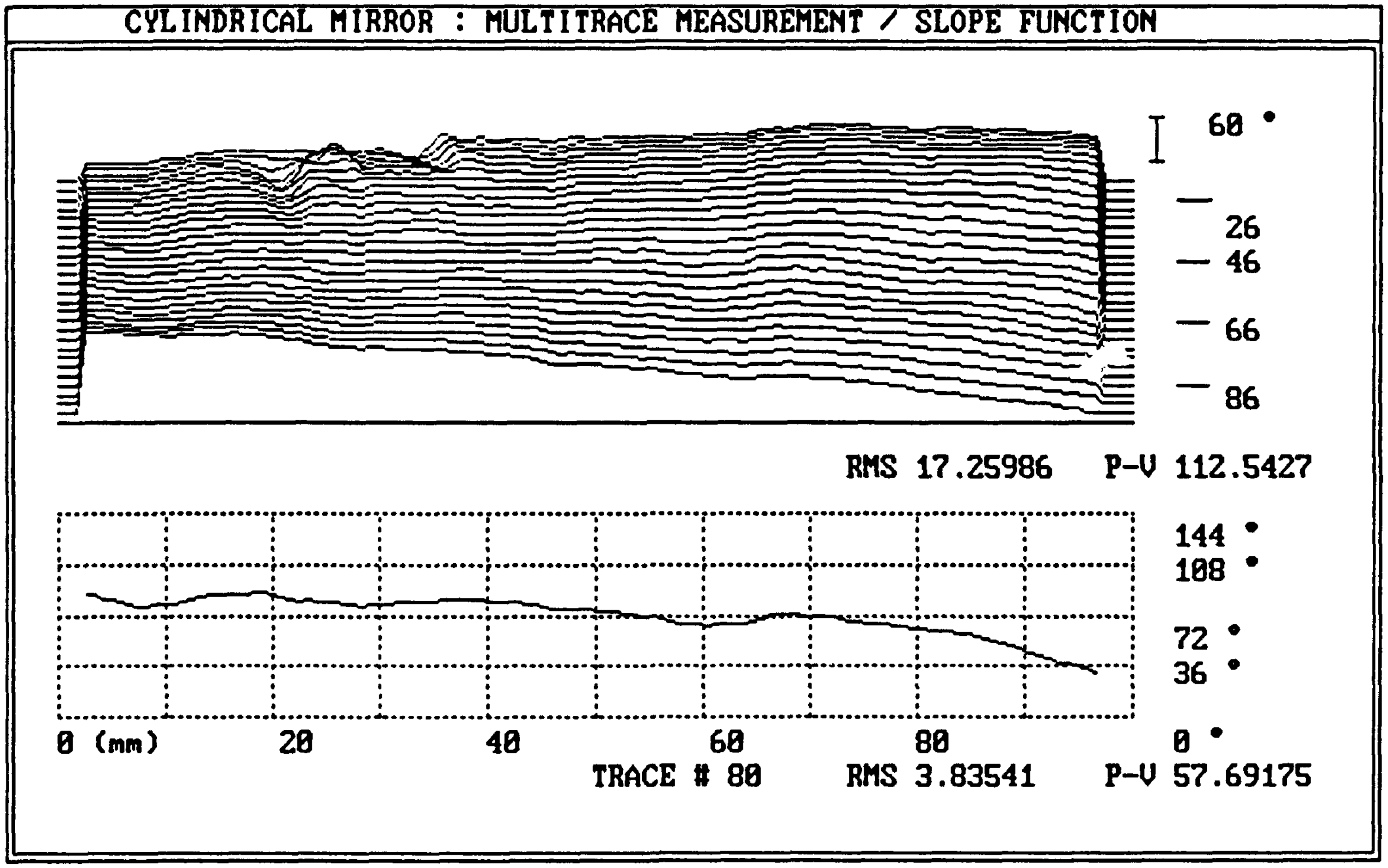
```
RMS 17.25986    P-V 112.5427
TRACE # 55      RMS 4.622995    P-V 37.36874
```

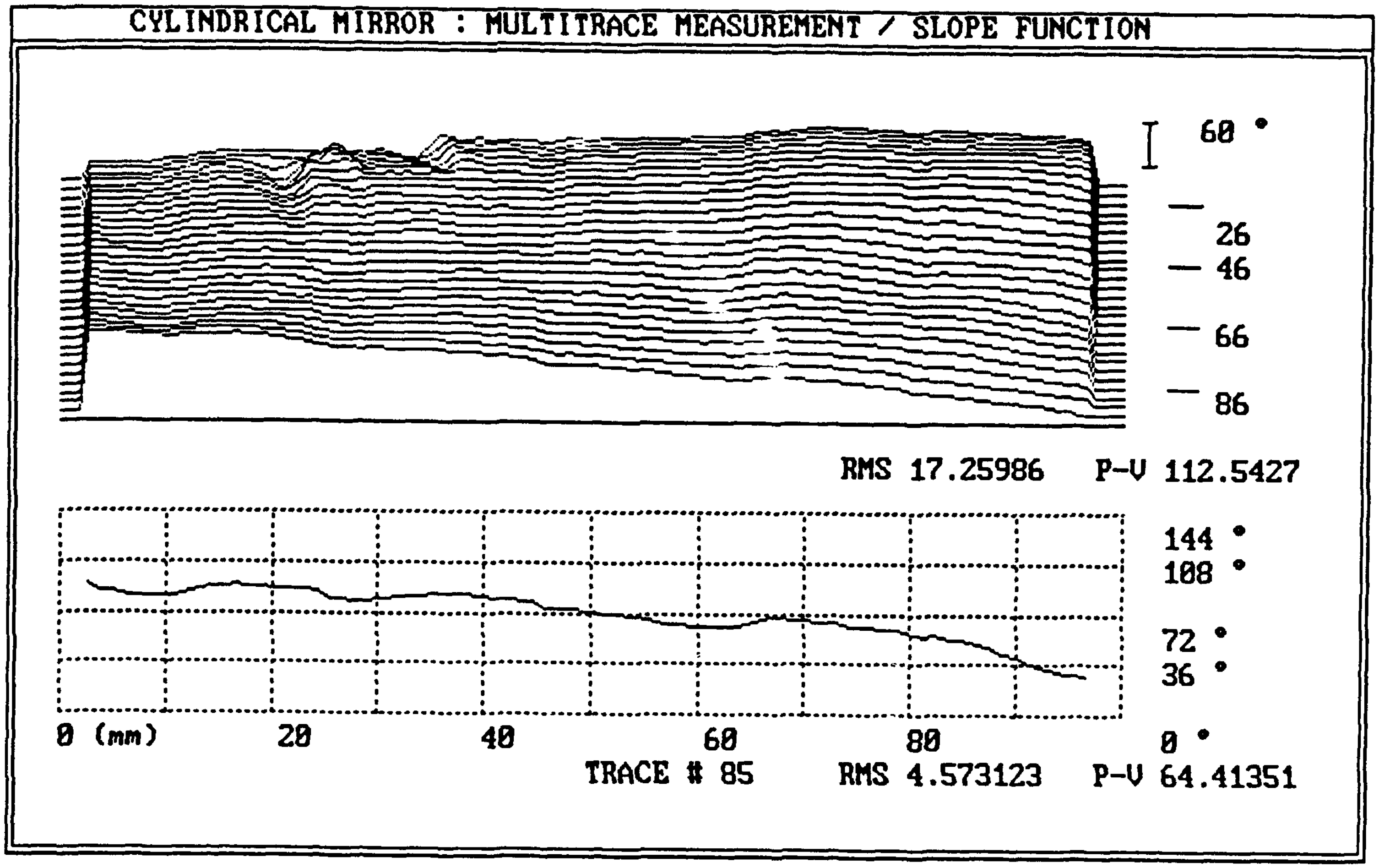

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986 P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 60
RMS 3.246115 P-V 31.79525

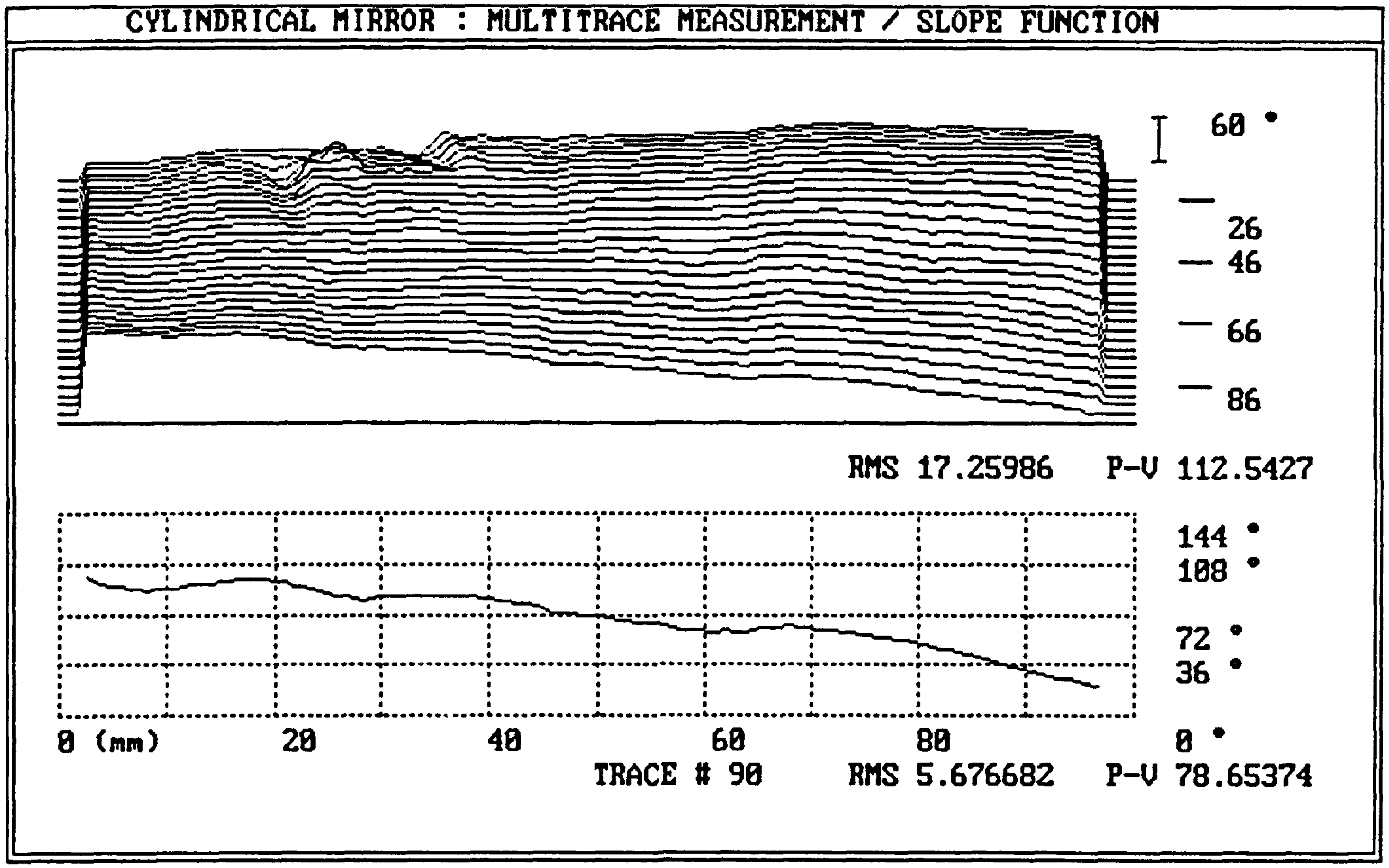

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986
P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 65
RMS 2.661066
P-V 37.34784


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60
26
46
66
86
RMS 17.25986
P-V 112.5427
144
108
72
36
0
0 (mm)
20
40
60
80
TRACE # 70
RMS 2.678192
P-V 41.50981


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 •
26
46
66
86
RMS 17.25986 P-V 112.5427
144 •
108 •
72 •
36 •
0 •
0 (mm)
20
40
60
80
TRACE # 75
RMS 3.402546 P-V 51.30483

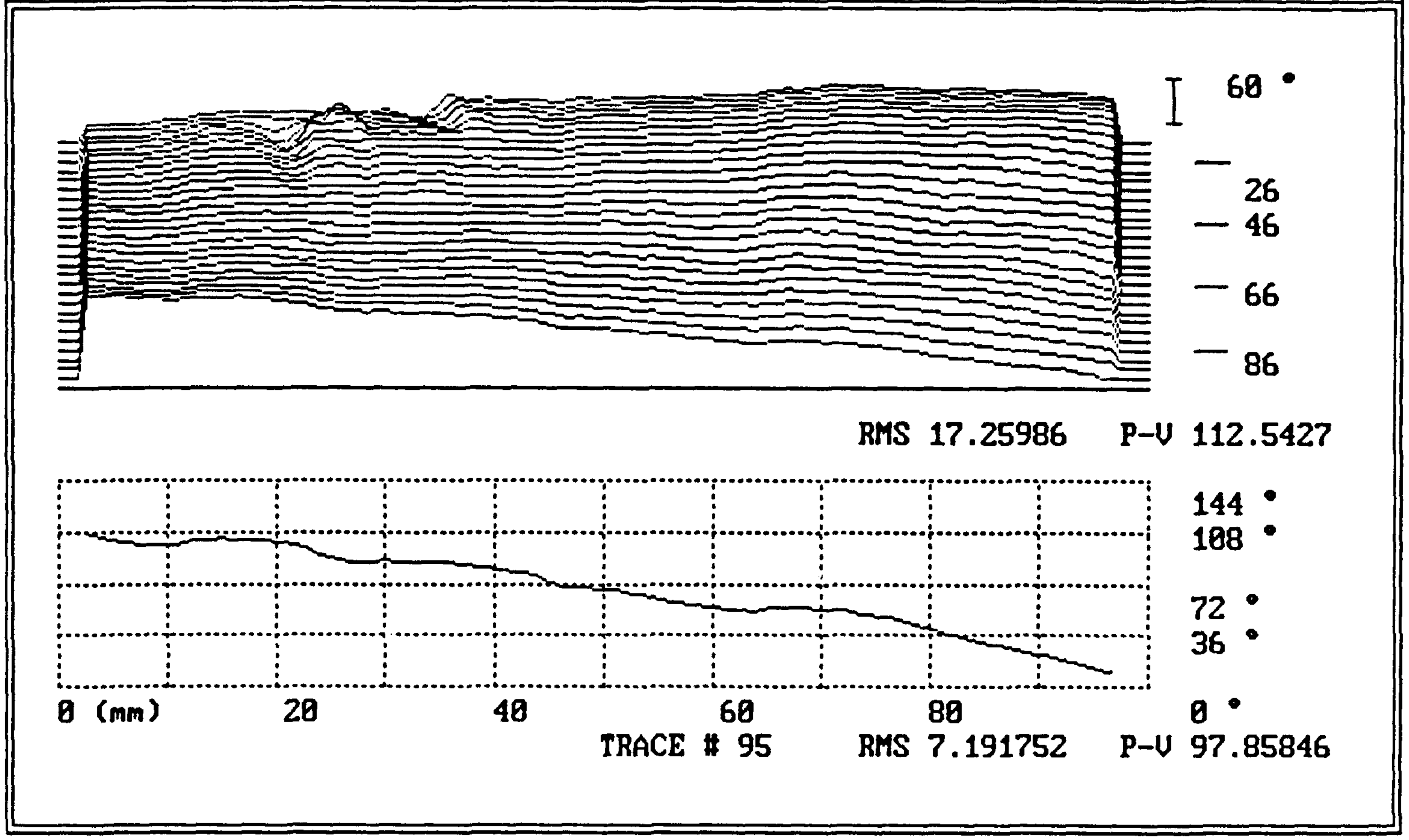

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 °
26
46
66
86
RMS 17.25986
P-V 112.5427
144 °
108 °
72 °
36 °
0 °
0 (mm)
20
40
60
80
TRACE # 80
RMS 3.83541
P-V 57.69175


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 °
26
46
66
86
RMS 17.25986 P-V 112.5427
144 °
108 °
72 °
36 °
0 °
0 (mm)
20
40
60
80
TRACE # 85
RMS 4.573123 P-V 64.41351



```
RMS 17.25986   P-V 112.5427
TRACE # 90   RMS 5.676682   P-V 78.65374
```

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION



RMS 17.25986 P-V 112.5427

TRACE # 95 RMS 7.191752 P-V 97.85846

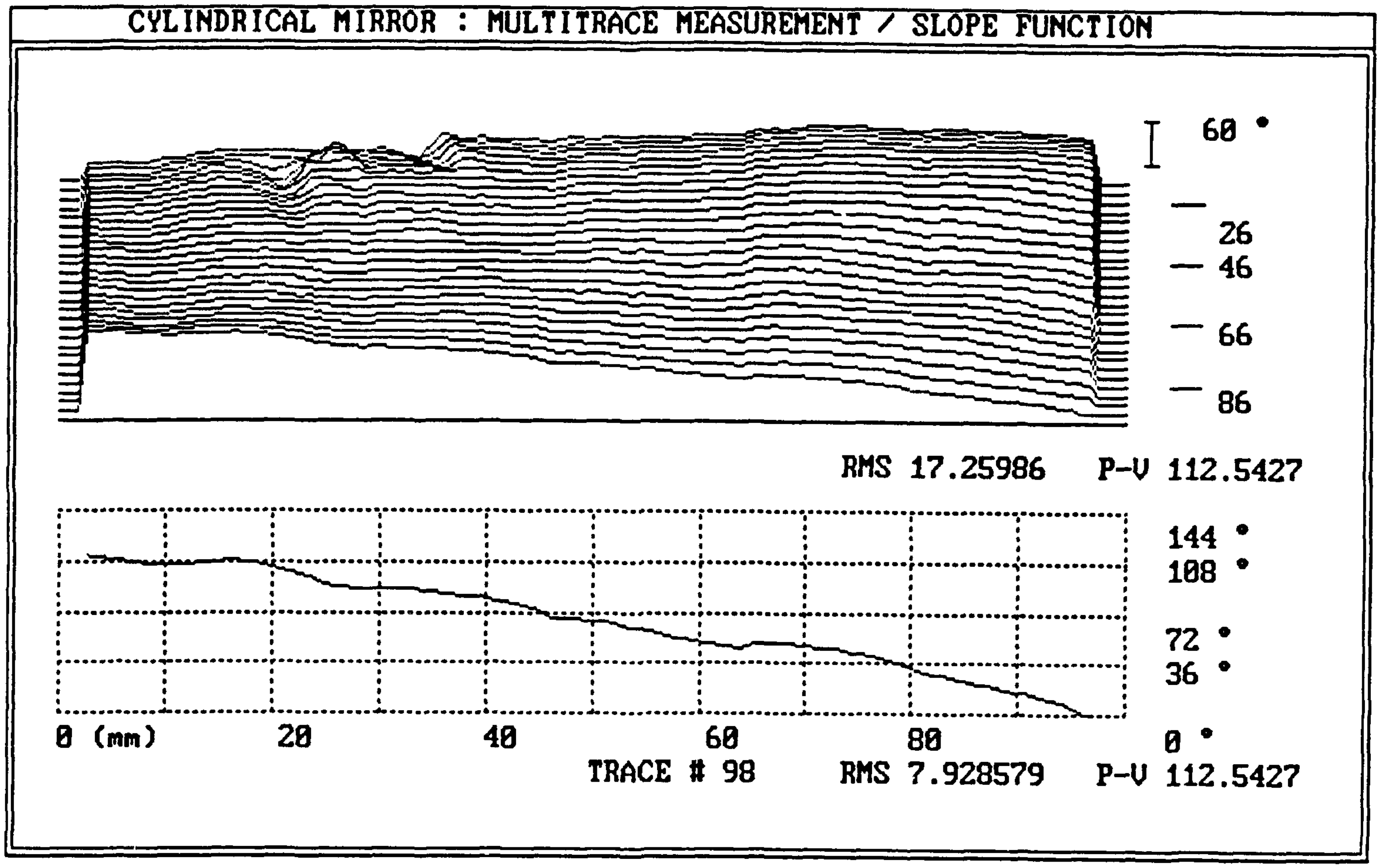
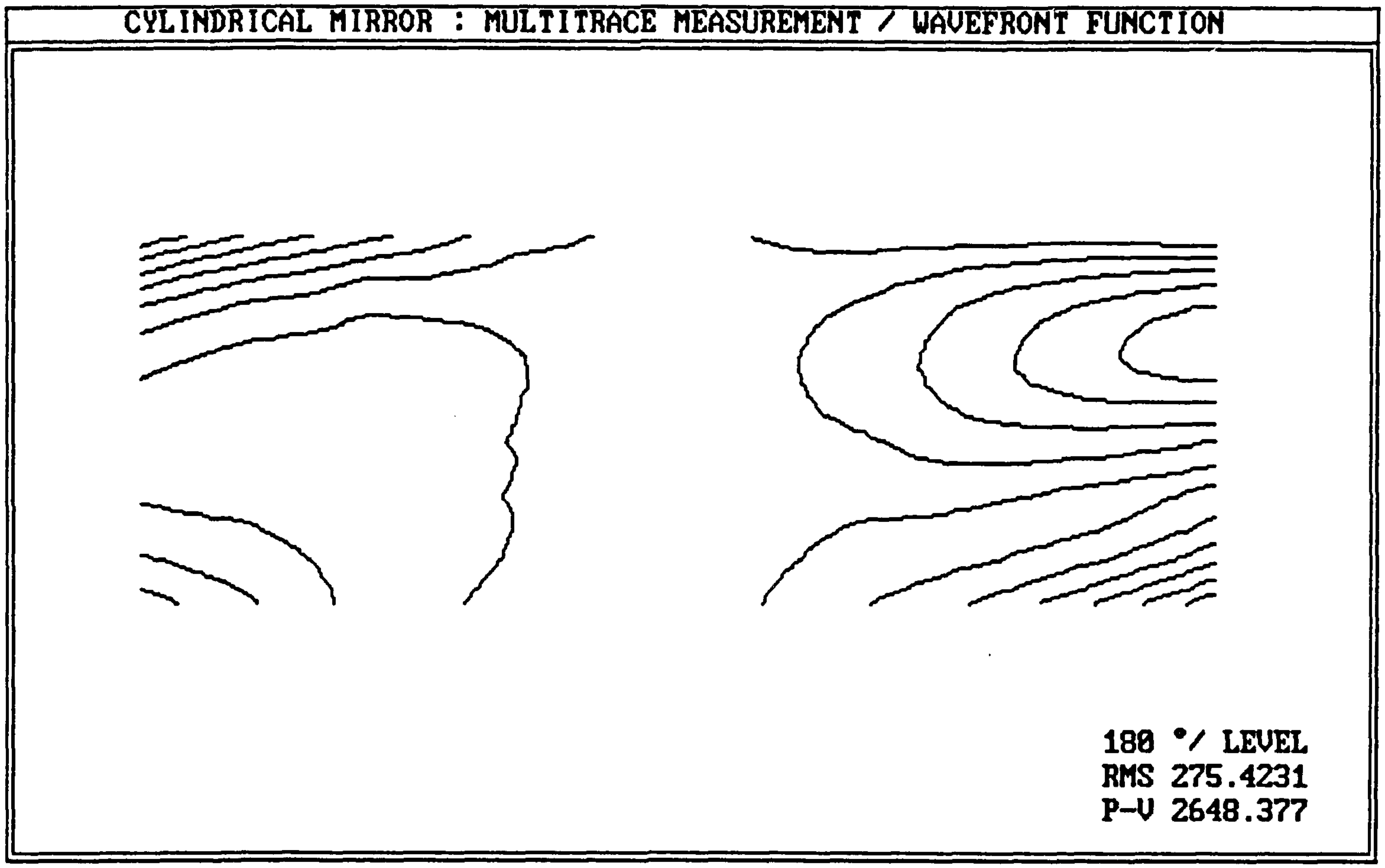
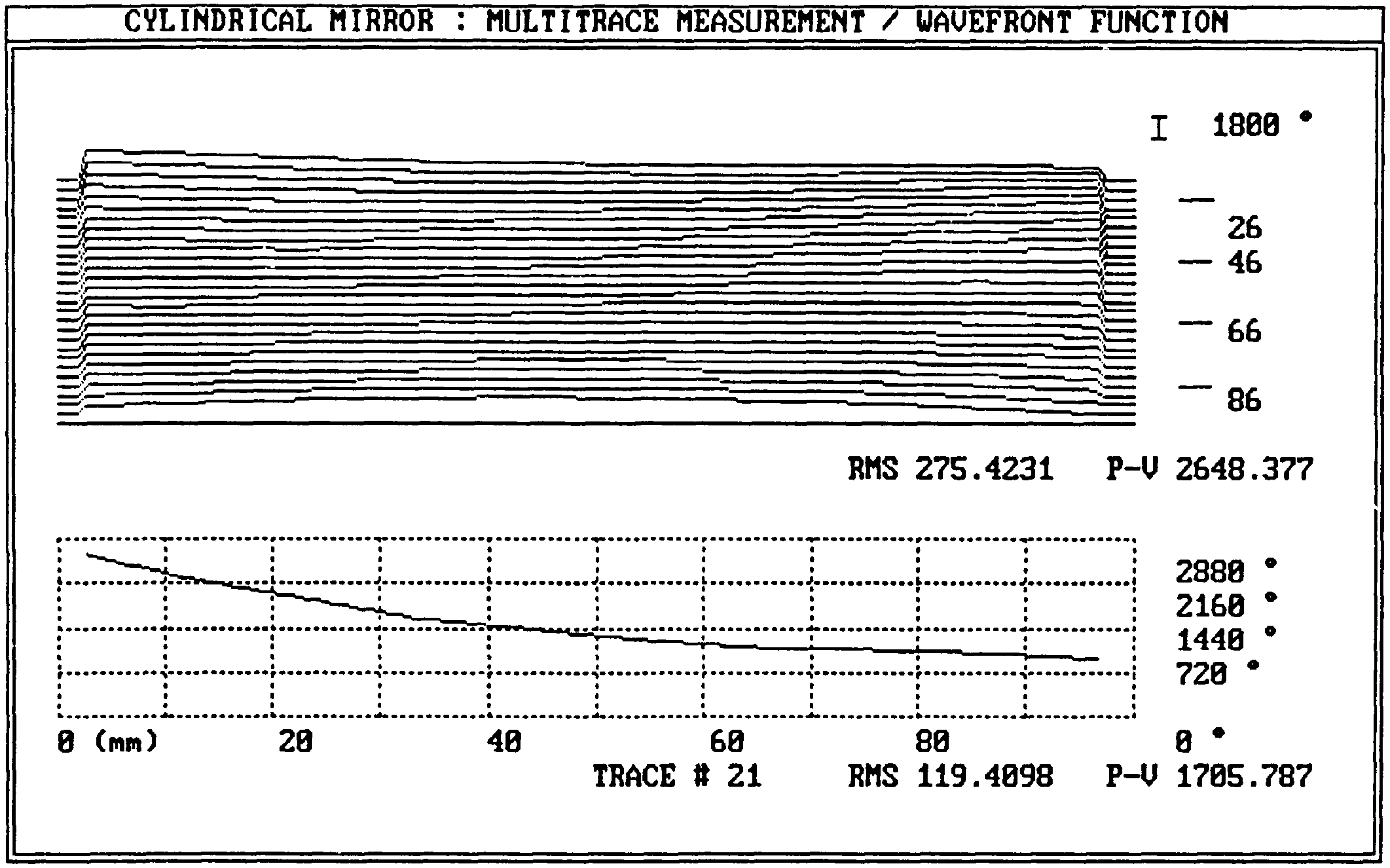
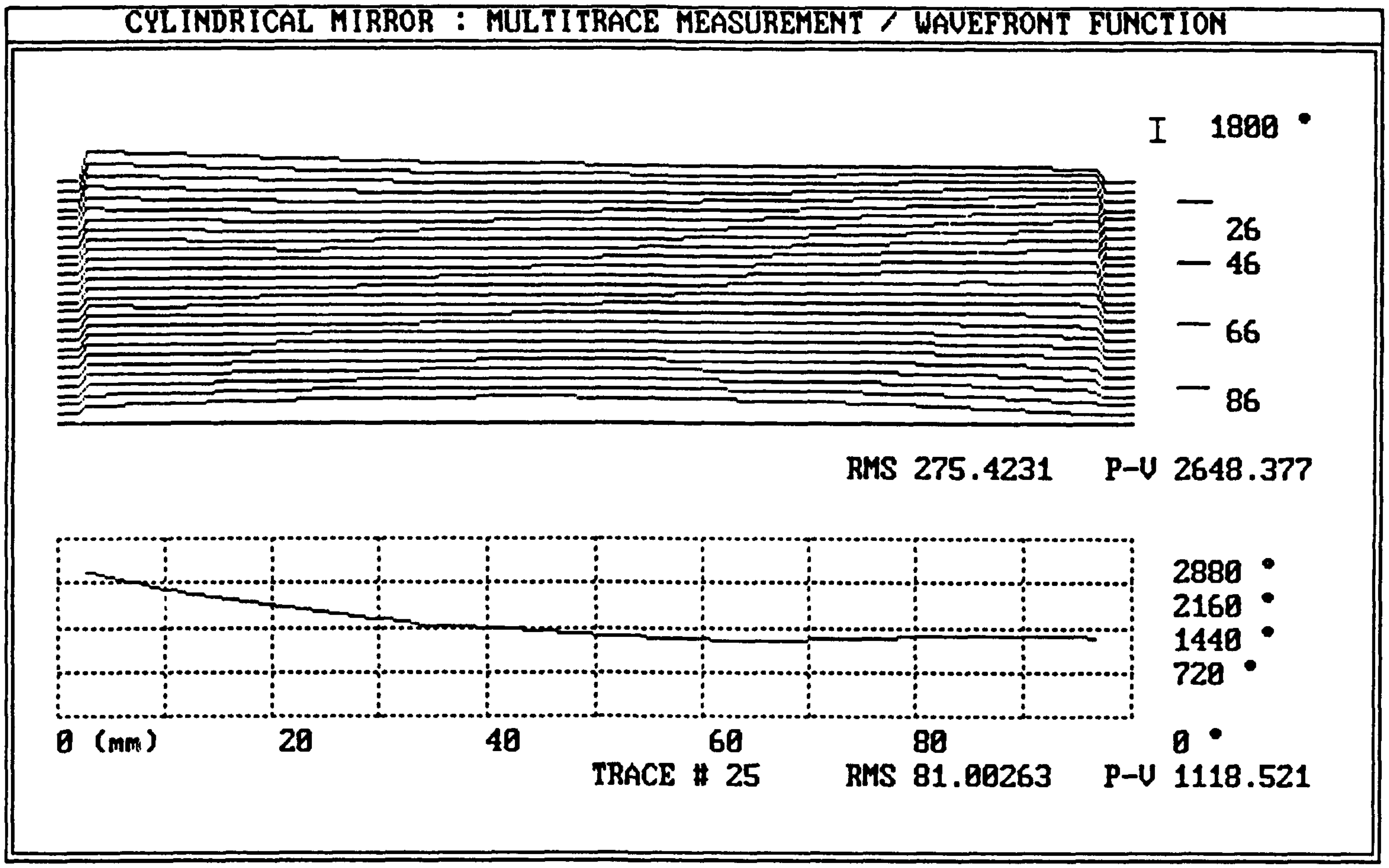
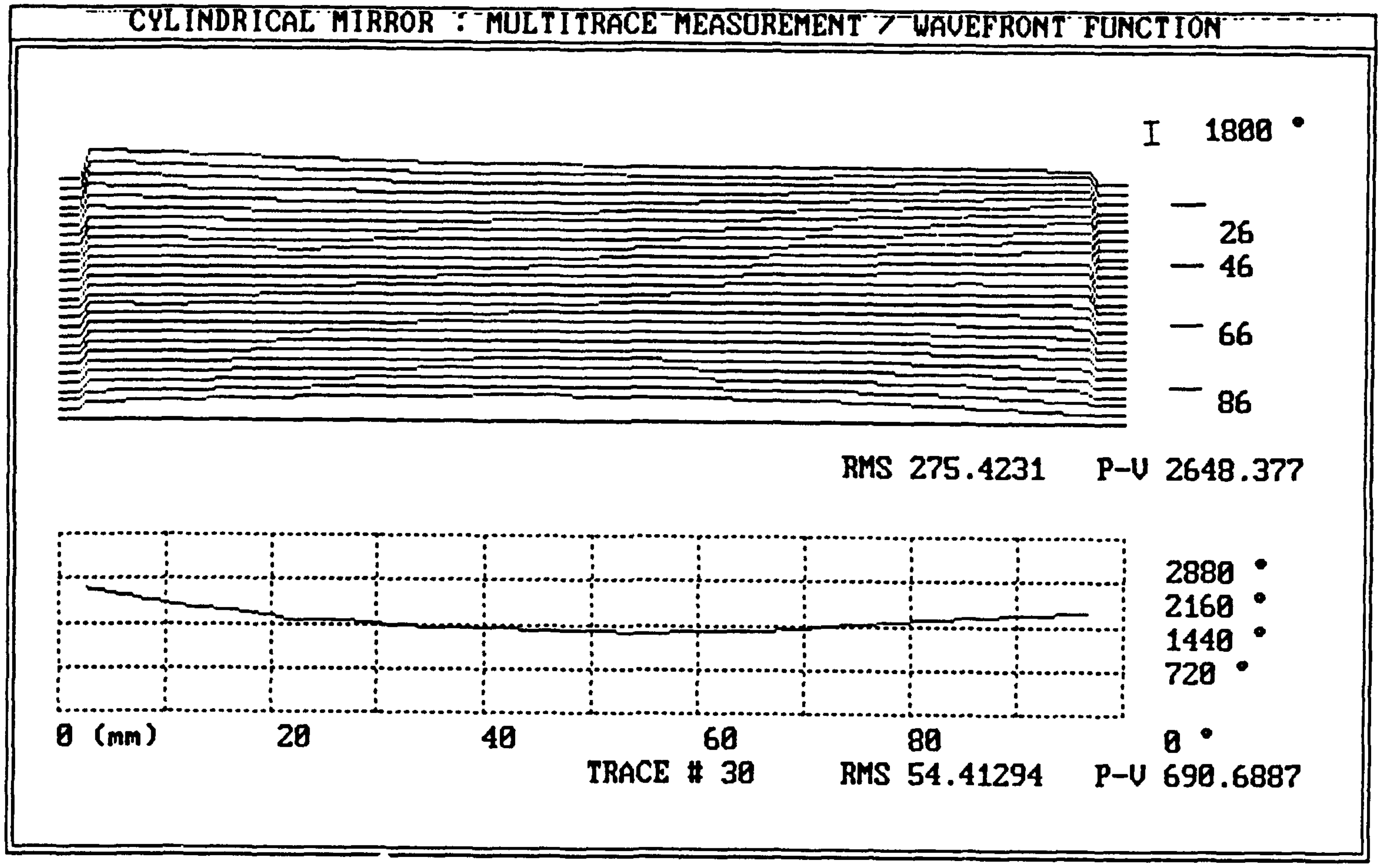

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / SLOPE FUNCTION
60 °
26
46
66
86
RMS 17.25986
P-V 112.5427
144 °
108 °
72 °
36 °
0 °
0 (mm)
20
40
60
80
TRACE # 98
RMS 7.928579
P-V 112.5427

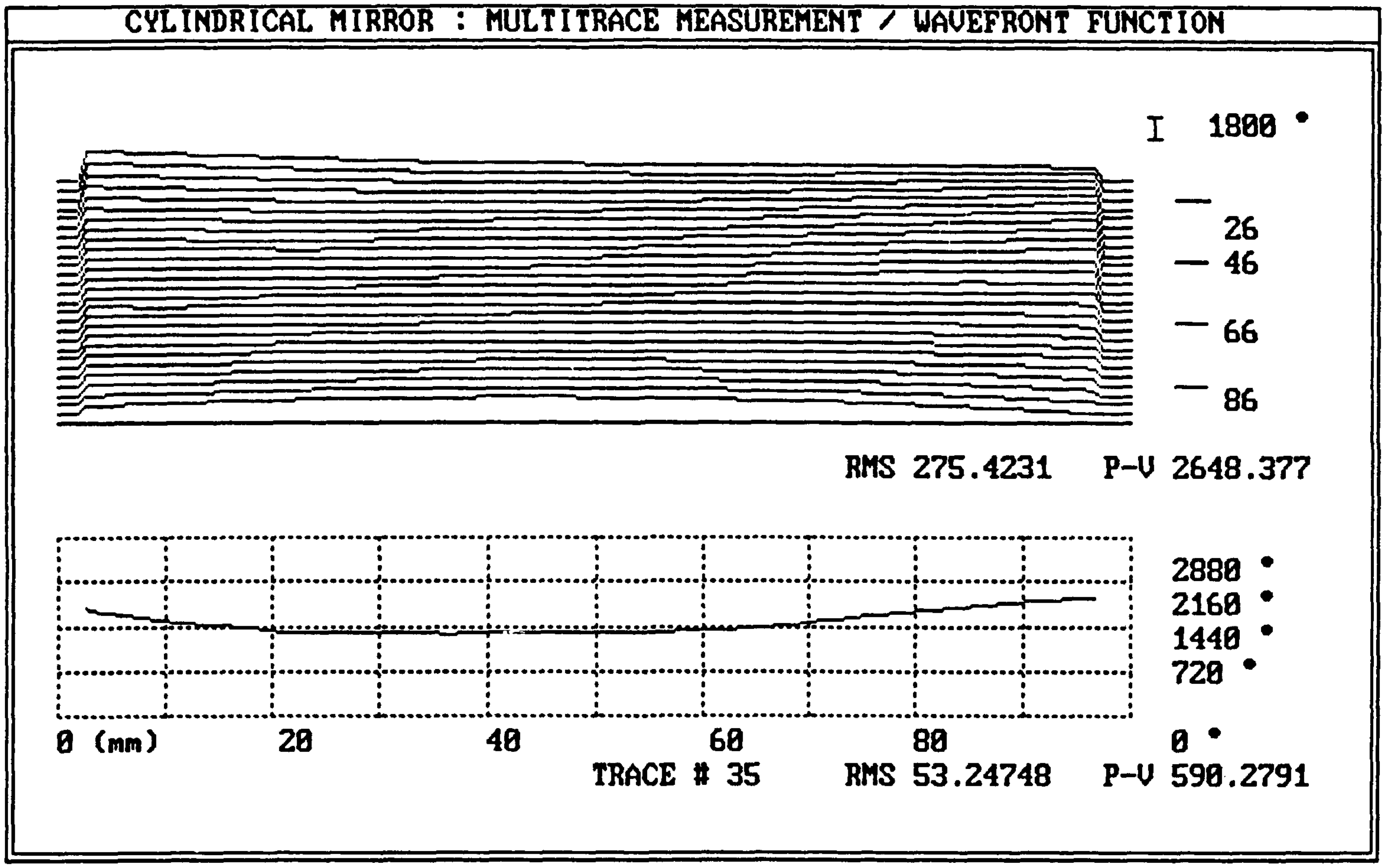
Figure 14: Results of Subaperture Testing (Wavefront Function)

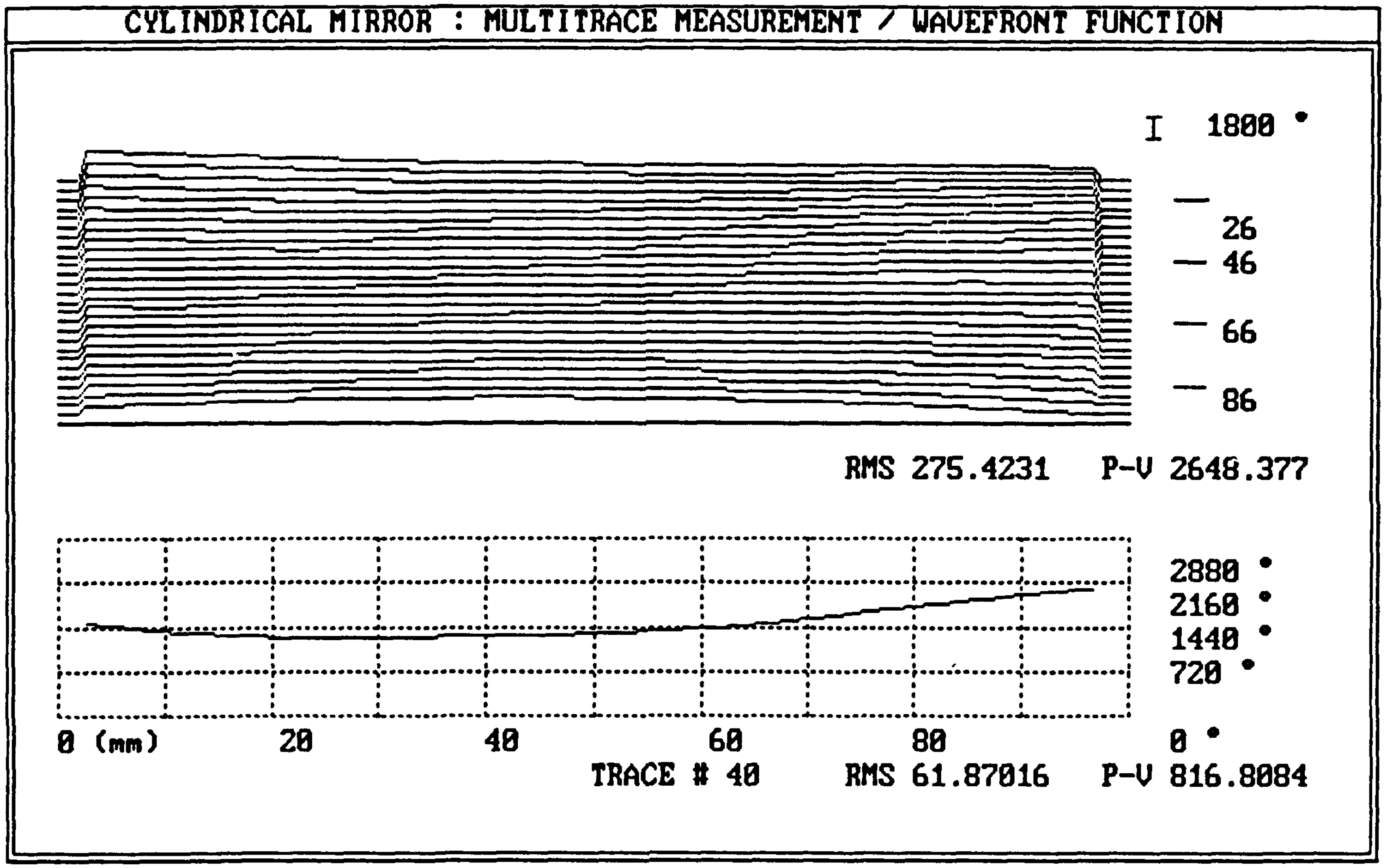

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
180 °/ LEVEL
RMS 275.4231
P-V 2648.377

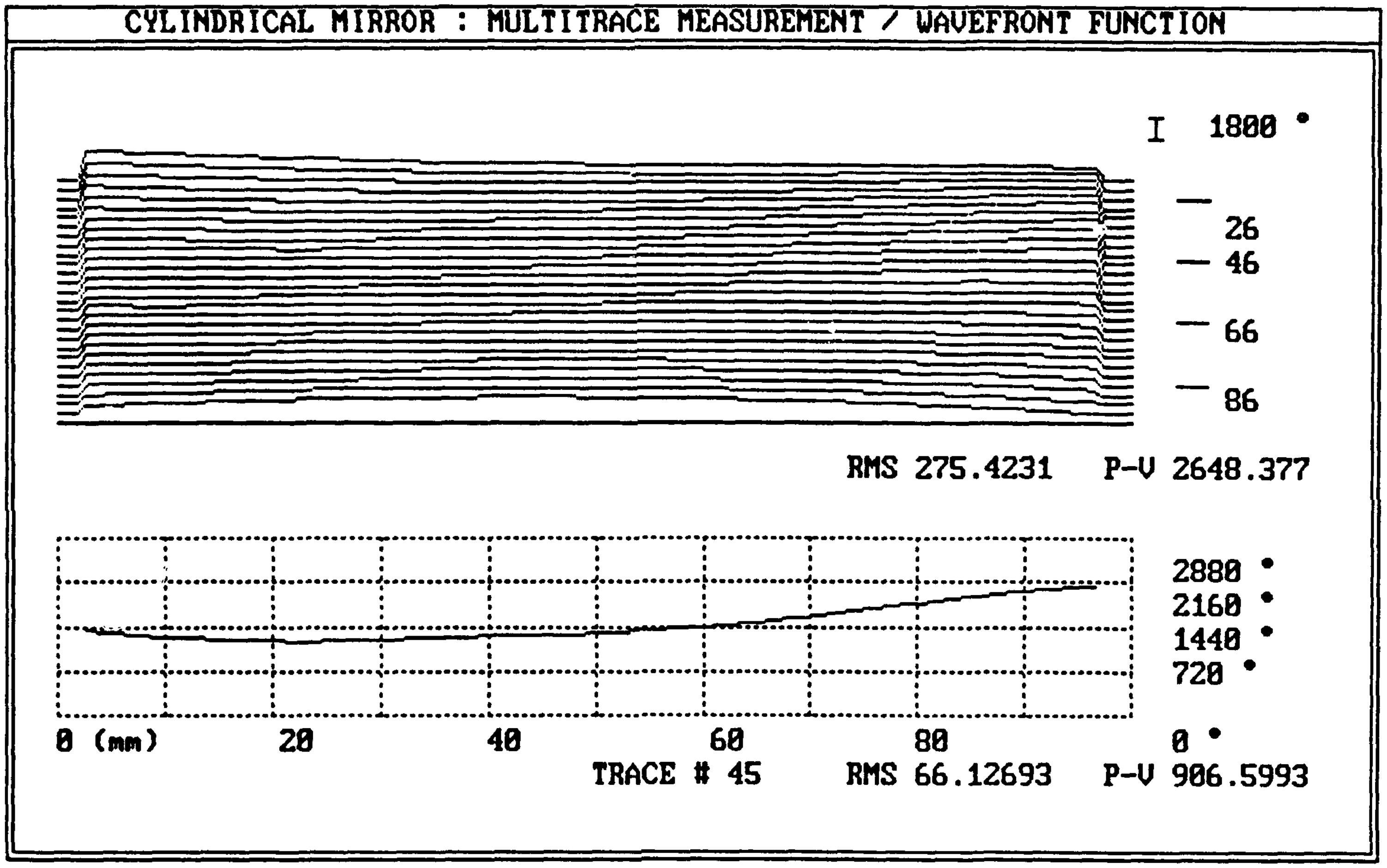

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 °
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 °
2160 °
1440 °
720 °
0 °
0 (mm)
20
40
60
80
TRACE # 21
RMS 119.4098 P-V 1705.787

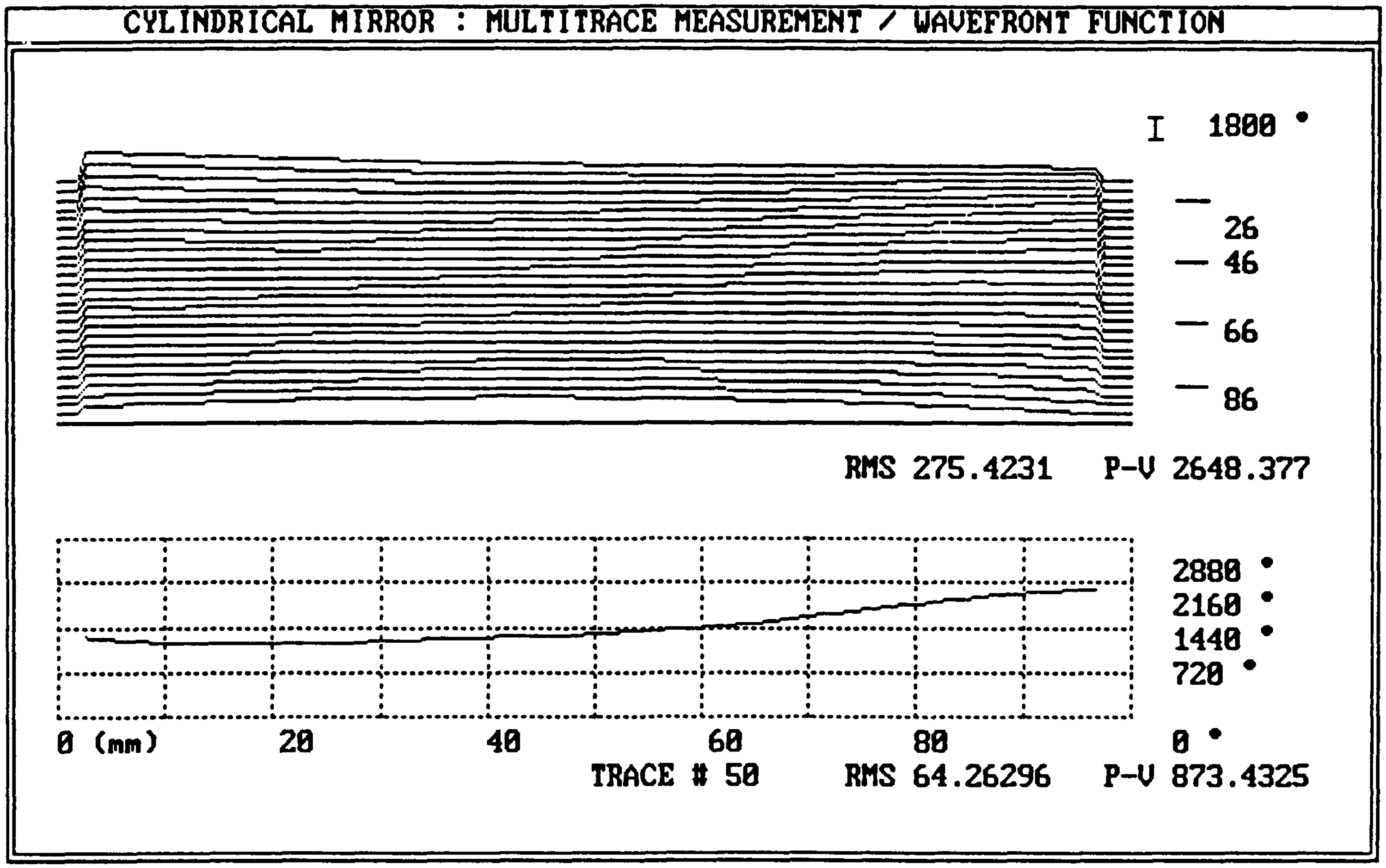

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 °
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 °
2160 °
1440 °
720 °
0 °
0 (mm)
20
40
60
80
TRACE # 25 RMS 81.00263 P-V 1118.521

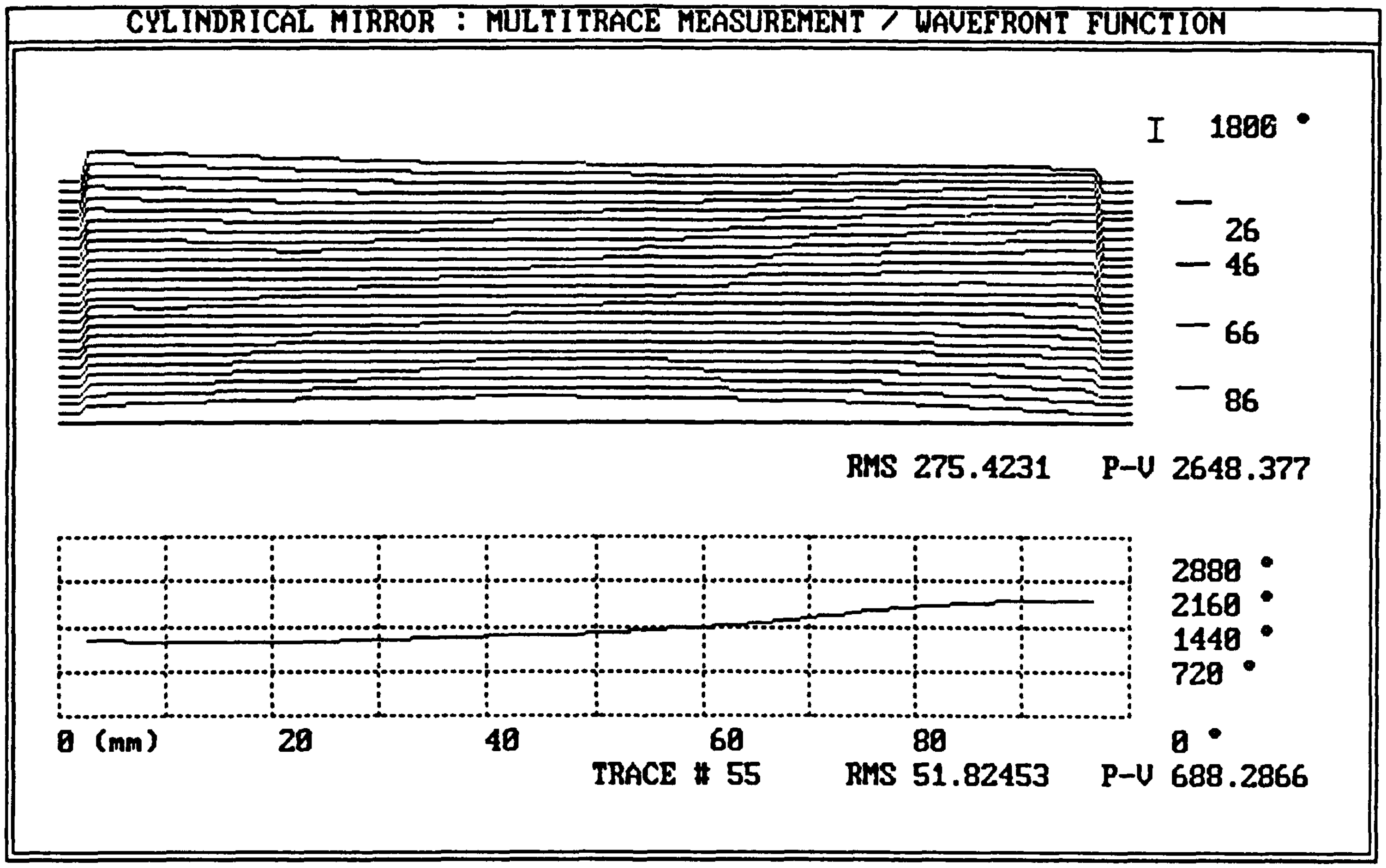

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 °
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 °
2160 °
1440 °
720 °
0 °
0 (mm)
20
40
60
80
TRACE # 30
RMS 54.41294 P-V 690.6887


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 •
26
46
66
86
RMS 275.4231
P-V 2648.377
2880 •
2160 •
1440 •
720 •
0 •
0 (mm)
20
40
60
80
TRACE # 35
RMS 53.24748
P-V 590.2791


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 °
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 °
2160 °
1440 °
720 °
0 °
0 (mm)
20
40
60
80
TRACE # 40 RMS 61.87016 P-V 816.8084


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 °
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 °
2160 °
1440 °
720 °
0 °
0 (mm)
20
40
60
80
TRACE # 45
RMS 66.12693 P-V 906.5993

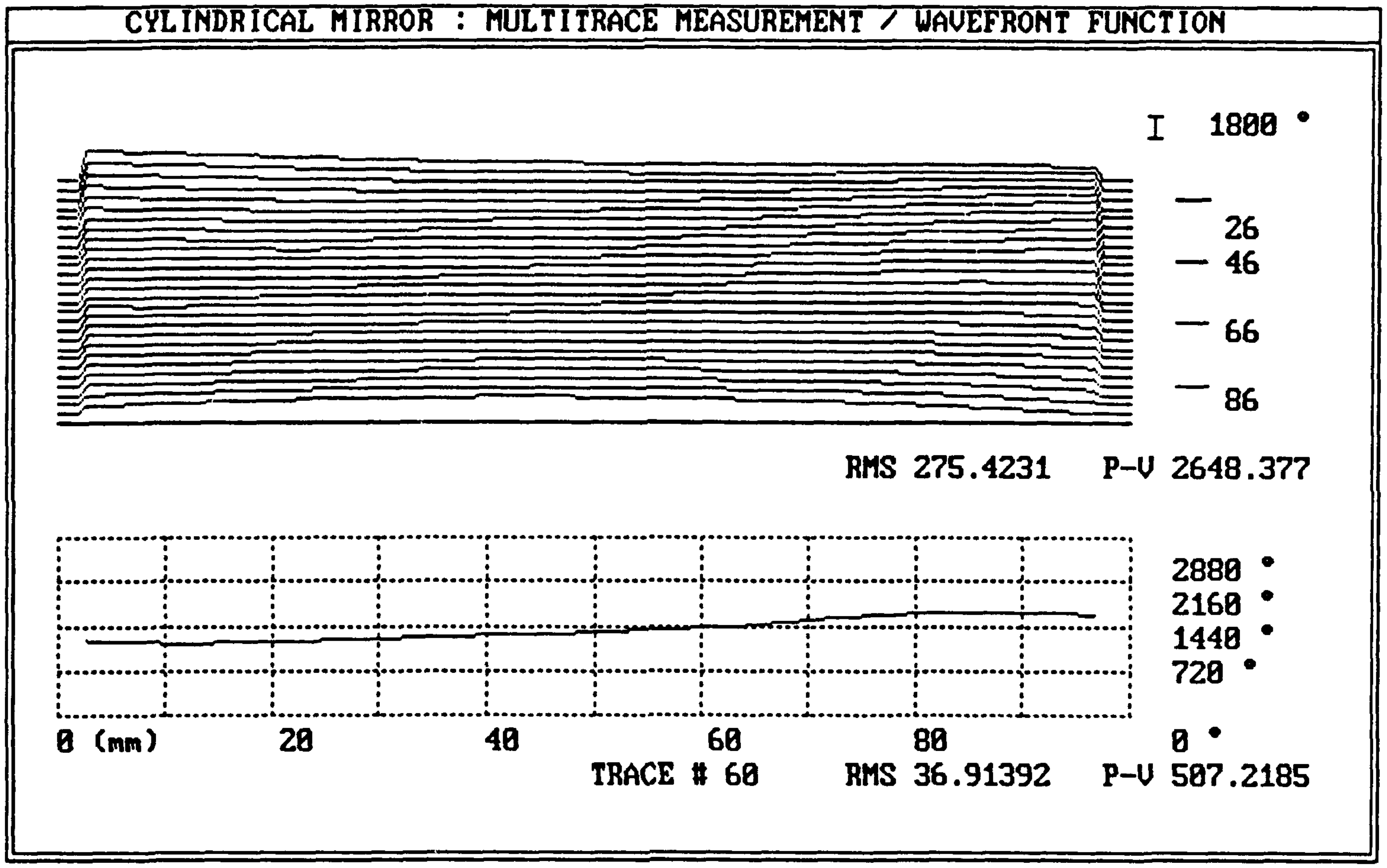

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800
26
46
66
86
RMS 275.4231 P-V 2648.377
2880
2160
1440
720
0
0 (mm)
20
40
60
80
TRACE # 50
RMS 64.26296 P-V 873.4325



```
CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION

                                   RMS 275.4231   P-V 2648.377

                     TRACE # 55    RMS 51.82453   P-V 688.2866
```

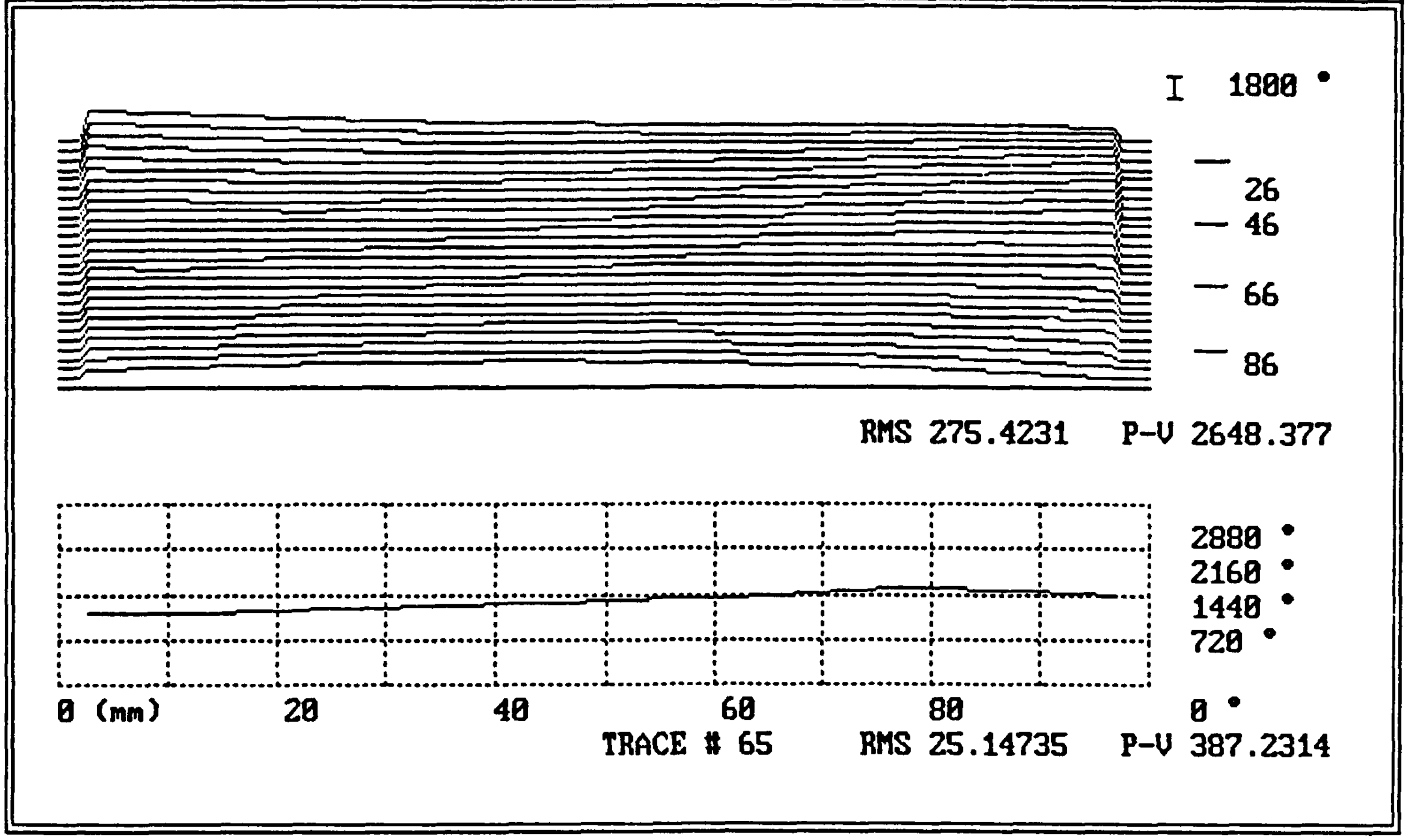


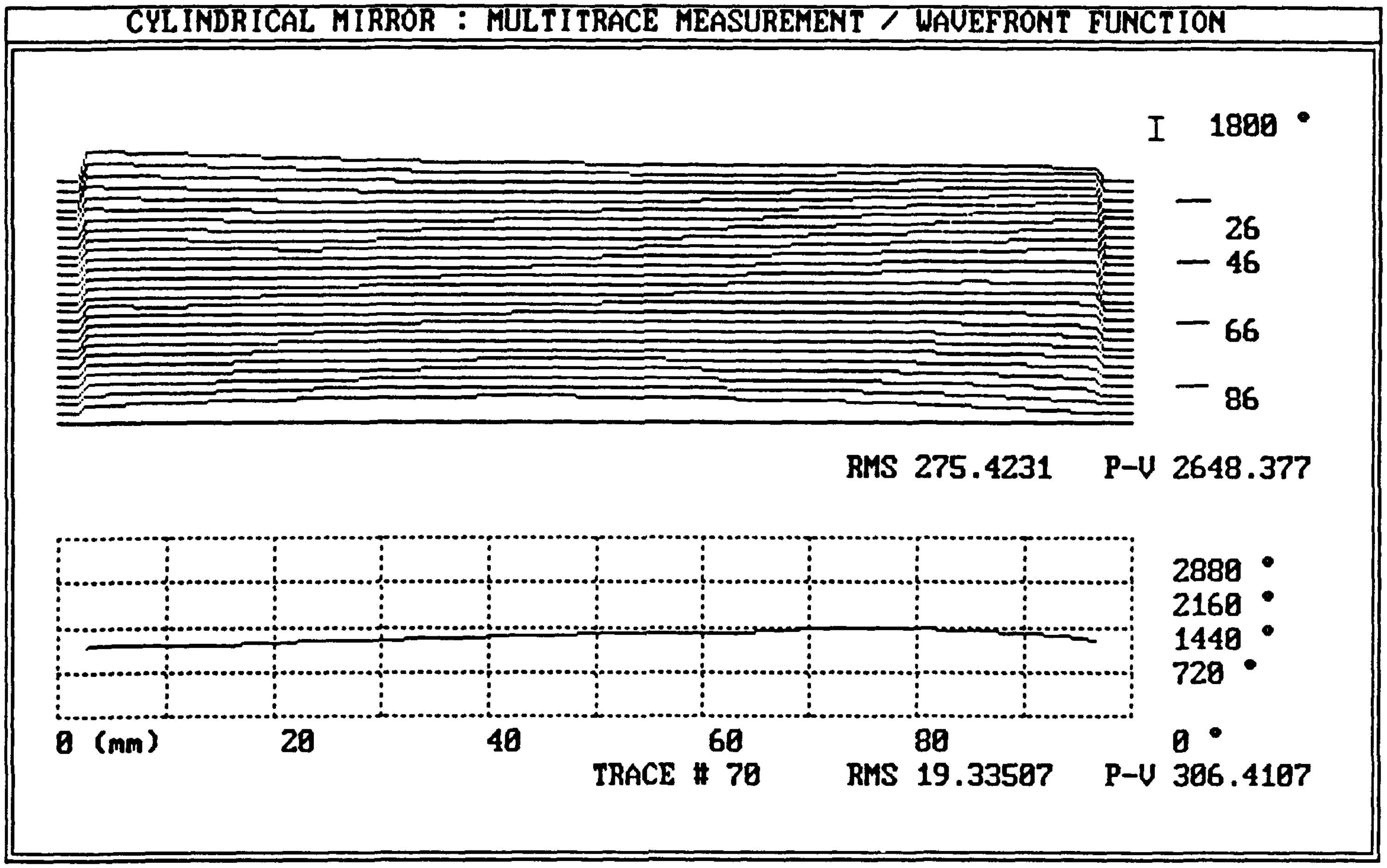
CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION

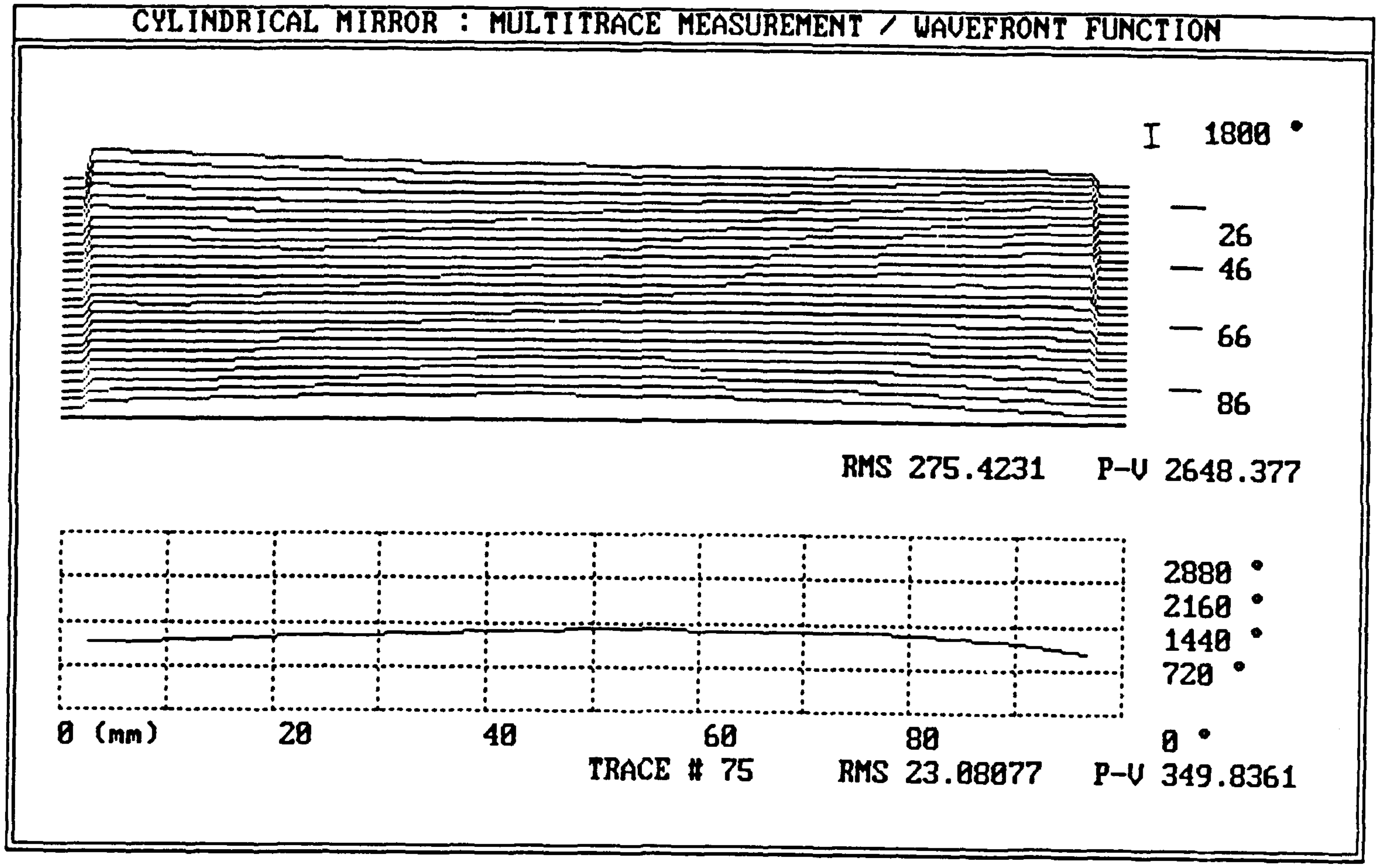
RMS 275.4231   P-V 2648.377

TRACE # 60   RMS 36.91392   P-V 507.2185

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION

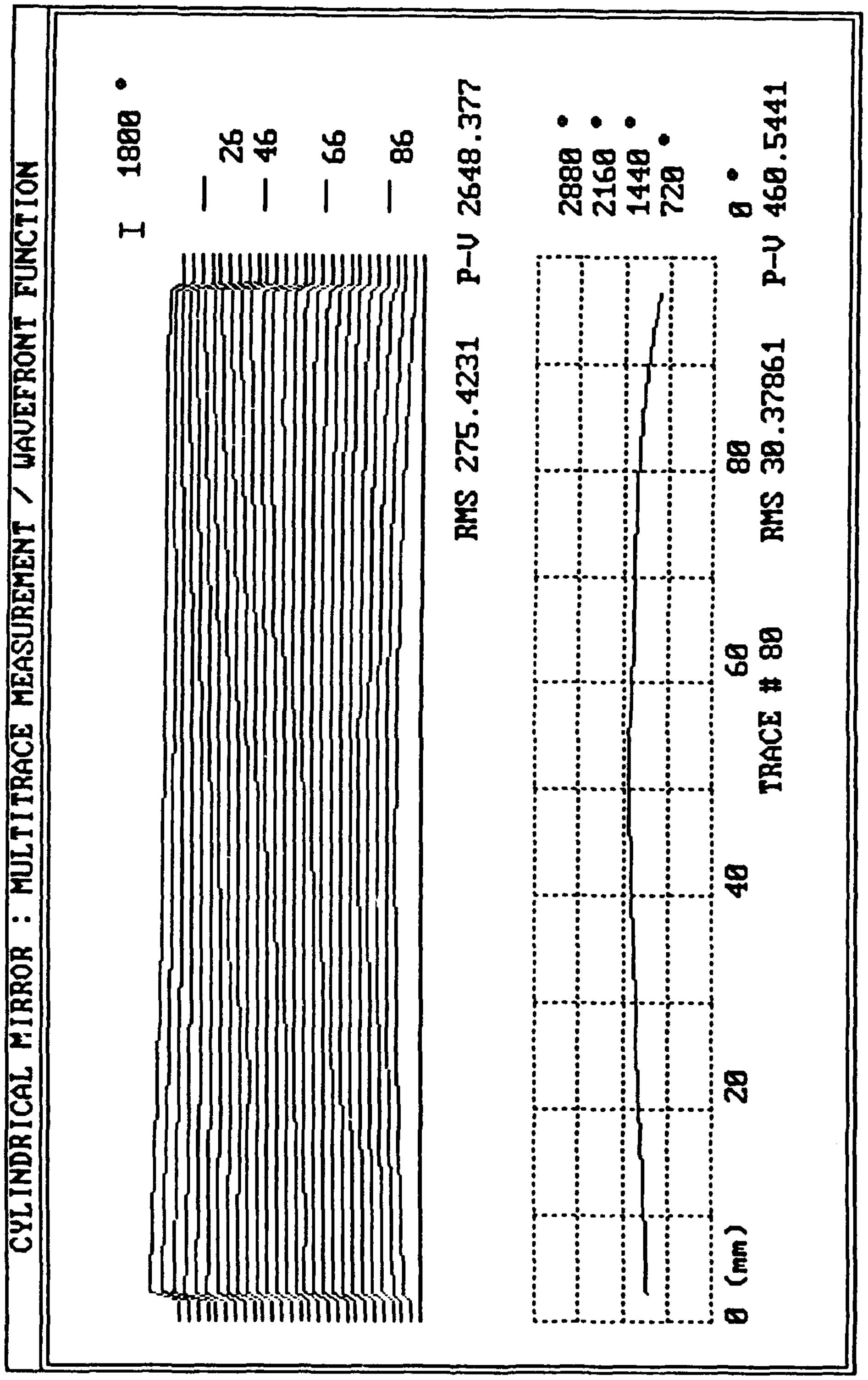


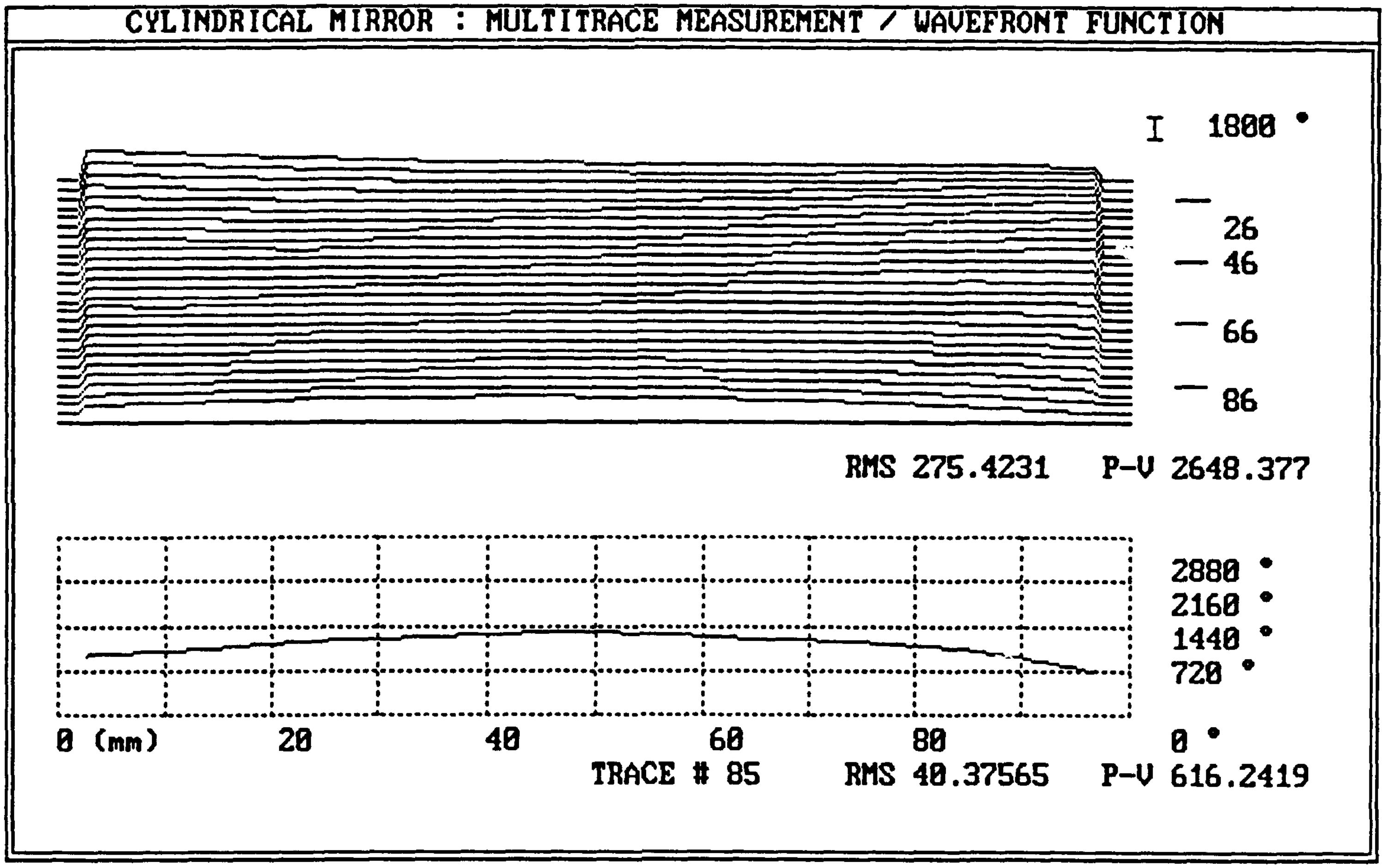

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 °
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 °
2160 °
1440 °
720 °
0 °
0 (mm)
20
40
60
80
TRACE # 70 RMS 19.33507 P-V 306.4107



RMS 275.4231 P-V 2648.377

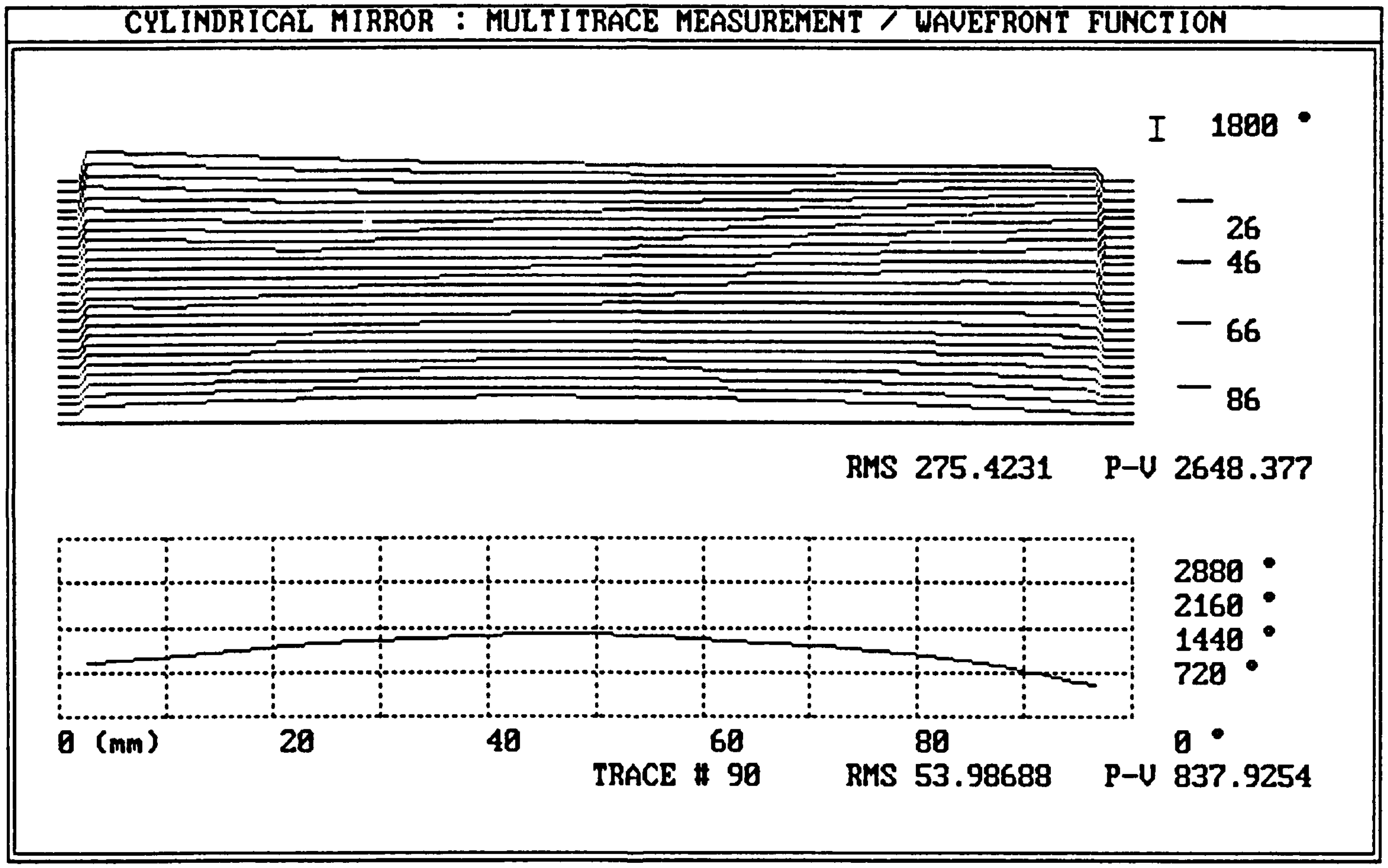
TRACE # 75 RMS 23.08077 P-V 349.8361

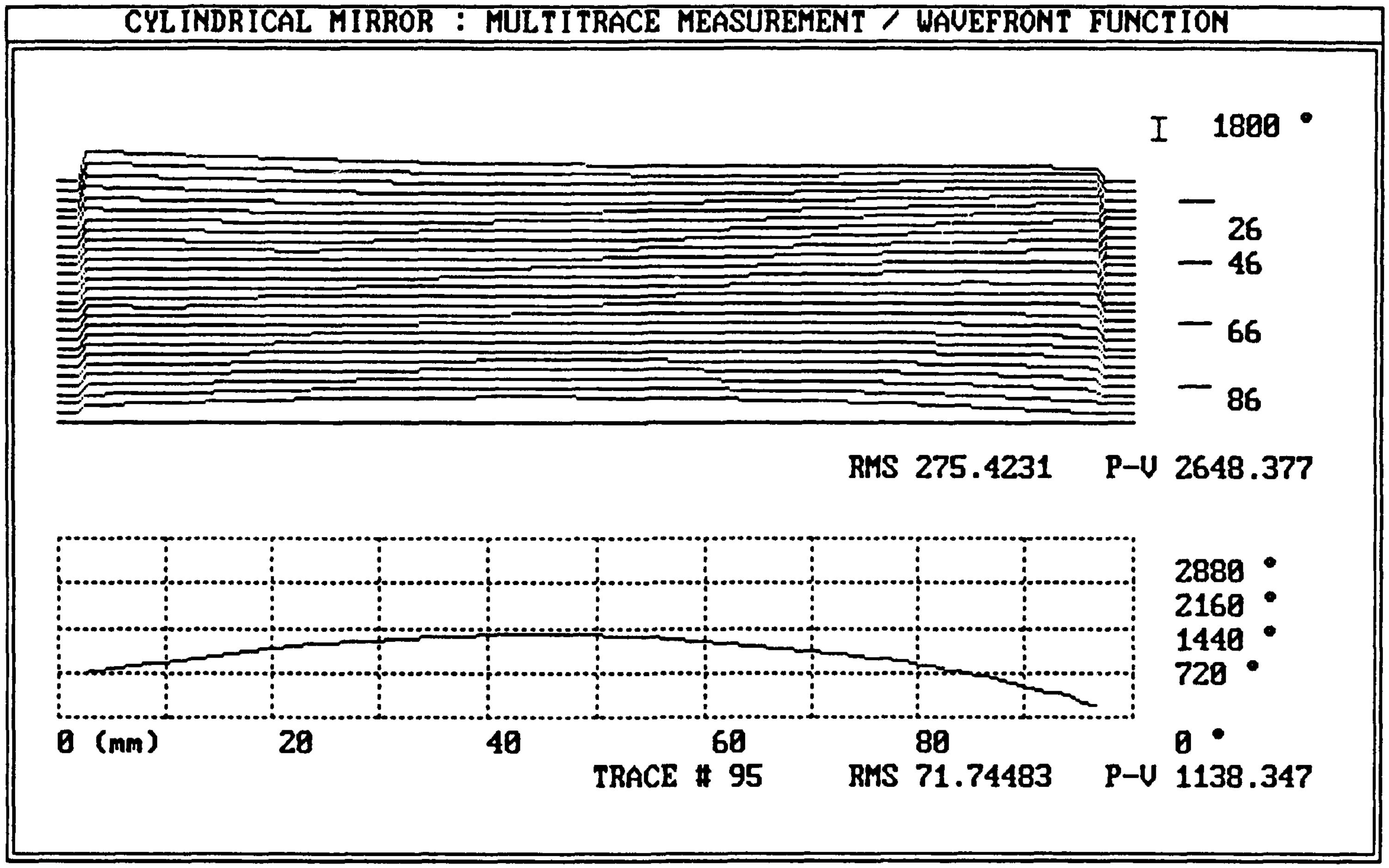

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800
26
46
66
86
RMS 275.4231 P-V 2648.377
2880
2160
1440
720
0
0 (mm)
20
40
60
80
TRACE # 80
RMS 30.37861 P-V 460.5441

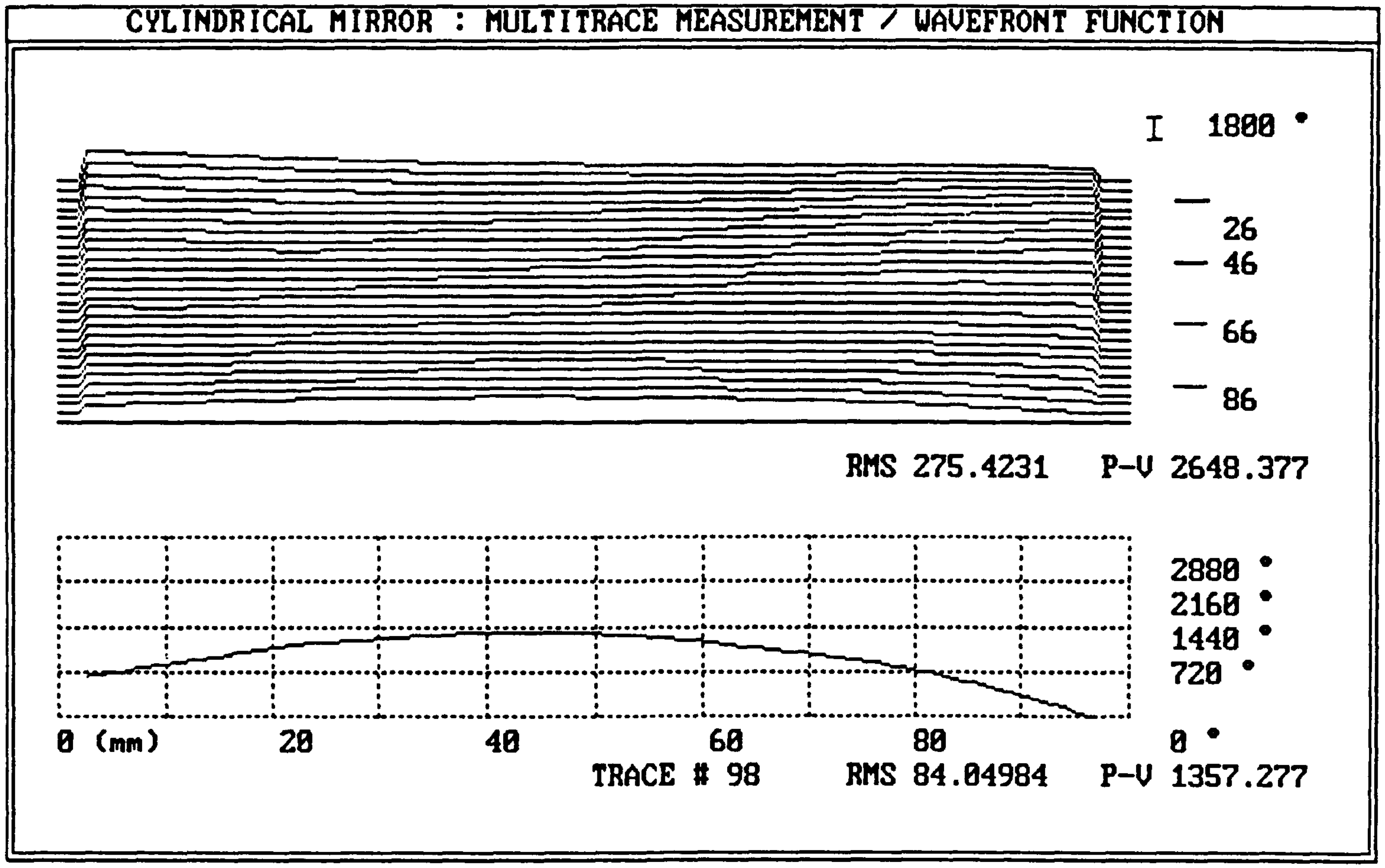

CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION

```
RMS 275.4231    P-V 2648.377

TRACE # 85    RMS 40.37565    P-V 616.2419
```


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 •
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 •
2160 •
1440 •
720 •
0 •
0 (mm)
20
40
60
80
TRACE # 90
RMS 53.98688 P-V 837.9254


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 °
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 °
2160 °
1440 °
720 °
0 °
0 (mm)
20
40
60
80
TRACE # 95
RMS 71.74483 P-V 1138.347


CYLINDRICAL MIRROR : MULTITRACE MEASUREMENT / WAVEFRONT FUNCTION
I 1800 •
26
46
66
86
RMS 275.4231 P-V 2648.377
2880 •
2160 •
1440 •
720 •
0 •
0 (mm)
20
40
60
80
TRACE # 98 RMS 84.04984 P-V 1357.277